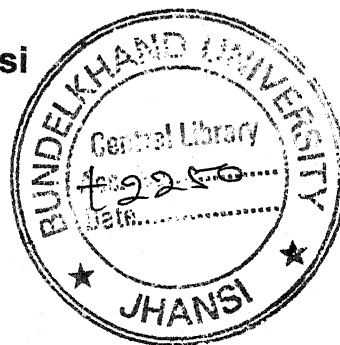


PERFORMANCE OF NEURAL NETWORK METHODS IN PATTERN RECOGNITION

**Thesis
Submitted to
Bundelkhand University, Jhansi**



**For the Degree of
Doctor of Philosophy
in
Computer Science and Engineering**

**By
Er. Yash Pal Singh**

**Under Supervision of
Dr. Rajeev Srivastava
Reader and Head
Department of Information Technology
Bundelkhand Institute of Engineering and Technology,
Jhansi**

Certificate

This is to certify that the thesis entitled "PERFORMANCE OF NEURAL NETWORK METHODS IN PATTERN RECOGNITION" submitted to the Bundelkhand University, Jhansi(U.P.) India for the award of the degree of Doctor of Philosophy in Computer Science and Engineering is a record of bonafide research work carried out by Yash Pal Singh under my Guidance and Supervision.

The work embodied in this thesis or a part were of has not been submitted for the award of any other degree. The Candidate has worked for more than 200 days to complete the research work under my supervision.



Supervisor

(Dr. Rajeev Srivastava)

Reader and Head

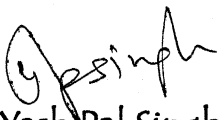
Department of Information Technology

Bundelkhand Institute of Engineering and Technology,

Jhansi.

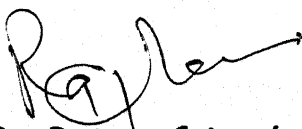
Declaration

I here by avow that the work presented in this thesis
**"PERFORMANCE OF NEURAL NETWORK
METHODS IN PATTERN RECOGNITION"** is entirely
my own work and there are no collaborators. It does not
contain any work for which any other university has awarded
degree.


(Yash Pal Singh)

Countersigned

Supervisor


(Dr. Rajeev Srivastava)

Reader and Head

Department of Information Technology

Bundelkhand Institute of Engineering and Technology,
Jhansi.

Acknowledgements

I feel greatly privileged in offering my sincere thanks and expressing deepest sense of gratitude and profound respects to my thesis supervisor Dr. Rajeev Srivastava, Reader and Head, Department of Information Technology, Bundelkhand Institute of Engineering and Technology, Jhansi, for his valuable guidance, constant encouragement, stimulating advice constructive and fruitful suggestions and above all significant contribution made by him, throughout my research work of Ph.D. without which this work would not have taken the present form. He always instilled in me a spirit of confidence and independent thinking and evaluation in the course of this investigation and preparation of the manuscript. I have all praise for him and will always go holding him in high esteem even throughout my educational career.

I don't have any appropriate words to express my heartiest thanks and gratitude to Prof. S.K.Awasthi, Director, Bundelkhand Institute of Engineering and Technology, Jhansi for his constant encouragement and moral support throughout my research work.

I am deeply thankful to respected, Prof. Kaliyan Singh, Ex- Vice Chancellor, Bundelkhand University, Jhansi and Ex-Member, Public Service Commission, Allahabad and Smt. Vimla Devi for their Blessings and encouragements.

From the depth of my heart I express my sincere thanks to Dr. Vikram Singh Yadav, Reader, Applied Science department, Bundelkhand Institute of Engineering and Technology, Jhansi and Mr Pappan Babu of Bundelkhand Institute of Engineering and Technology, Jhansi for their encouragements and support.

I am also thankful to Dr. Sultan Singh, Sr. Scientist IGRI, Jhansi and Dr. Meenakshi Singh, Head, Home Science Deptt., Bundelkhand University, Jhansi.

I am thankful to Mr. & Mrs. Hemant Kumar, Mr. & Mrs. Vedvrat and Mr. & Mrs. Indravesha Arya and Mr. Ramniwas for their support.

On this Occasion, I can't forget to pay my sincere thanks to my beloved wife, Dr. Pratibha Arya, Assistant Professor, Home Science Deptt., Bundelkhand University, Jhansi and my children Devshree and Shauryan, for their constant love, encouragement and patience during this time consuming work.


Er. Yash Pal Singh

Contents

Chapter 1: Introduction	1
1.1 Application of Pattern Technology	2
1.2 The Neural Network Algorithms used in Pattern recognition	6
1.3 Problem Statement	7
1.4 Organization of Thesis	7
 Chapter 2: Fundamentals of Artificial Neural networks	 9
2.1 Biological Neurons and their Artificial Models	9
2.1.1 Biological Neurons	11
2.1.2 Artificial Neurons	12
2.1.3 Single Layer Artificial Neural Network	16
2.1.4 Multi Layer Artificial Neural Network	17
2.2 Terminologies, Notations and Representations of Artificial Neural Networks	18
2.3 Training of Artificial Neural Networks	19
 Chapter 3: Neural Network Algorithms	 24
3.1 Bidirectional Associative Memory (BAM)	24
3.1.1 Memory Architecture	25
3.1.2 Association Encoding and Decoding	28
3.1.3 Algorithm	28
3.1.4 Storage Capacity	29
3.1.5 Results	30
3.1.6 Merits and Demerits	31
3.1 Hopfield Autoassociative Memory	31
3.2.1 Algorithm	32
3.2.2 Storage Capacity	32
3.2.3 Results	33

3.2.4 Merits and Demerits	33
3.3 Feature Recognition Neural Network	36
3.3.1 Algorithm	36
3.3.2 Results	38
3.3.3 Merits and Demerits	38
3.4 Hamming Network and MAXNET	41
3.4.1 Algorithm	47
3.4.2 Storage Capacity	48
3.4.3 Results	49
3.4.4 Merits and Demerits	49
3.5 Adaptive Resonance Theory 1 (ART1)	49
3.5.1 Algorithm	51
3.5.2 Storage Capacity	53
3.5.3 Results	54
3.5.4 Merits and Demerits	54
3.6 Back Propagation Algorithm	56
3.6.1 Algorithm	57
3.6.2 Results	58
3.6.3 Merits and Demerits	59
3.7 Quickprop	59
3.7.1 Results	60
3.7.2 Merits and Demerits	61
3.8 Kohonen Self-organizing Map	62
3.8.1 Algorithm	62
3.8.2 Results	64
3.8.3 Merits and Demerits	65
3.9 Neocognitron	65
3.9.1 Algorithm	65
3.9.2 Storage Capacity	67
3.9.3 Results	67
3.9.4 Demerits	69

Chapter 4: Different Analysis Criterion	70
4.1 Introduction	70
4.2 Effect of Noise in Weight on Various Algorithm	71
4.2.1 When weights are Varied Randomly	71
4.2.2 By adding Constant number to all the weight of network	74
4.2.3 Effect of Noise in Inputs on Various algorithms	75
4.2.4 Loss of Connections	76
4.2.5 Missing Information	78
4.2.6 Adding Information	79
Chapter 5: Conclusions	83
5.1 General Conclusions	83
5.2 Recommendations for Future Work	88
Appendix A	89
Appendix B	119
References	121

Chapter 1: Introduction

Chapter 1: Introduction

Pattern recognition techniques are associated a symbolic identity with the image of the pattern. In this work we will analyze different neural network methods in pattern recognition. This problem of replication of patterns by machines (computers) involves the machine printed patterns. The pattern recognition is better known as optical pattern recognition. Since it deals with recognition of optically processed patterns rather than magnetically processed [1] ones. Though the origin of character recognition can be found as early as 1870, it first appeared as an aid to the visually handicapped and the first successful attempt was made by the Russian scientist Tyurin in 1900 [2]. The modern version of optically character recognition (OCR) appeared in the middle of the 1940's with the development of digital computers. Thenceforth it was realized as a data processing approach with application to the business world. The principal motivation for the development of OCR systems is the need to cope with the enormous flood of paper such as bank cheques, commercial forms, govt. records, credit card imprints and mail sorting generated by the expending technological society. OCR machines have been commercially available since the middle of 1950's. Since then extensive research has been carried out and a large no. of technical papers and reports has been published by the various researchers in the area of pattern recognition. Several books have been published on optical character [3-11] Russian scientists Tyurin recognition. Also special issues and reports on the topic have repeatedly appeared in the proceedings of the international joint conferences on pattern recognition and of the international systems, Man and

cybernetics conferences. Research works also appear in the various other conferences such as British conferences on pattern recognition, and Scandinavian conferences on image analysis.

1.1 Application of Pattern Technology

Pattern recognition has many practical applications.

- Use by blind people- as reading aid using photo sensor and tactile simulators, and as a sensory aid with sound output [12-13].
- Use as a telecommunication aid for deaf [14].
- Use in postal department – for postal address reading as a reader for handwritten and printed postal codes [15-17].
- Use in publishing industry [18], and as a reader for data communication terminal [19].
- For direct processing of documents- as a multipurpose document reader for large scale data processing, as a microfilm reader data input system, for high speed data entry, for changing graphics in to a computer readable form, as electronic page reader to handle large volumes of mail [20-21].
- For use in customer billing as in telephone exchange billing system [22], order data logging [23], automatic fingerprint identification [24], as an automatic inspection system-I.C. mask inspection and defect detection in microcircuits [25], as a credit card scanner in credits personal identification systems.
- For business application- financial business application like cheques sorting strategy optimization [26, 27].
- For digital barcode reading, and as a handwritten analyzers- for automatic writer recognition and signature verification [28, 29].

- For shorthand transcription, and electronic package industries and reading characters stamped on metallic parts.

A neural network is a processing device, whose design was inspired by the design and functioning of human brain and their components. There is no idle memory containing data and programmed, but each neuron is programmed and continuously active.

Neural network has many applications. The most likely applications for the neural networks are (1) Classification (2) Association and (3) Reasoning. One of the applications of neural networks is in the field of pattern recognition. Pattern recognition is a branch of artificial intelligence concerned with the classification or description of observations. Its aim is to classify patterns based on either a priori knowledge or on the features extracted from the patterns. For pattern recognition there should be techniques to describe a large no of similar structures of the same category and allowing distinct description among different category. It is expected that if the features people use to recognize pattern are properly giving to the pattern recognition algorithms of neural network, then it should perform as human.

Pattern recognition is the recognition or separation of one particular sequence of bits or pattern from other such patterns. Pattern recognition [PR] applications have been varied, and so also the associated data structures and processing paradigms [30]. In the course of time, four significant approaches to PR have evolved. These are:

Statistical pattern recognition: Here, the problem is posed as one of composite hypothesis testing, each hypothesis pertaining to the premise, of the datum having originated from a particular class; or as one of regression from the space of measurements to the space of classes. The statistical

methods for solving the same involve the computation other class conditional probability densities, which remains the main hurdle in this approach. The statistical approach is one of the oldest, and still widely used [31].

Syntactic pattern recognition: In syntactic pattern recognition, each pattern is assumed to be composed of sub-pattern or primitives strung together in accordance with the generation rules of a grammar characteristic of the associated class. Class identifications accomplished by way of parsing operations using automata corresponding to the various grammars [32, 33]. Parser design and grammatical inference are two difficult issues associated with this approach to PR and are responsible for its somewhat limited applicability.

Knowledge-based pattern recognition: This approach to PR [34] is evolved from advances in rule-based system in artificial intelligence (AI). Each rule is in form of a clause that reflects evidence about the presence of a particular class. The sub-problems spawned by the methodology are:-

1. How the rule-based may be constructed, and
2. What mechanism might be used to integrate the evidence yielded by the invoked rules?

Neural Pattern Recognition: Artificial Neural Network (ANN) provides an emerging paradigm in pattern recognition. The field of ANN encompasses a large variety of models [35], all of which have two important characteristics:

1. They are composed of a large number of structurally and functionally similar units called neurons usually connected various configurations of by weighted links.
2. The ANN's model parameters are derived from supplied I/O paired data sets by an estimation process called training.

ANN's had been developed as a class of trainable model-free estimators in an attempt to emulate the working of naturally intelligent systems (e.g. The Human brain).

A neural network is a processing device, whose design was inspired by the design and functioning of the human brain and their components. There is no idle memory containing data and programs, but each neuron is preprogrammed and continuously active.

Neural Networks have many applications. The most likely applications for neural network are classification, association and reasoning, one of the applications of neural networks in the field of pattern recognition. Pattern recognition is a branch of artificial intelligence concerned with the classification or description of observations.

Its aim is to classify patterned based on either a prior knowledge or on the features extracted from the pattern recognition, there should be techniques to describe a large number of similar structures of the same category and allowing distinct descriptions among different category [36]. It is expected that if the future people use to recognize pattern are properly given to the pattern recognition algorithms of neural network, then it should perform as human

Methodology: There are many neural network algorithms for the pattern recognition. Various algorithms differ in their learning mechanism. Learning can be either supervised or unsupervised. In supervised learning, the training set contains both inputs and required responses. After training the network we should get the response equal to target response. Unsupervised classification learning is based on clustering of input data. There is no prior information about input's membership in a particular class. The characteristics of the patterns and a history of training are used to assist the

network in defining classes. This unsupervised classification is called clustering.

The characteristics of the neurons and initial weights are specified based upon the training method of the network. The pattern sets is applied to the network during the training. The pattern to be recognized are in the form of vector whose elements is obtained from a pattern grid. The elements are either 0 and 1 or -1 and 1. In some of the algorithms, weights are calculated from the pattern presented to the network and in some algorithms weights are initialized. The network acquires the knowledge from the environment. The network stores the patterns presented during the training in another way it extracts the features of pattern. As a result of this, the information can be retrieved later.

1.2 The Neural Network Algorithms Used In Pattern Recognition

We use different neural network algorithms in pattern recognition as follows:

1. Bidirectional Associative Memory (BAM)
2. Hopfield Autoassociative Memory
3. Feature Recognition Neural Network (FRNN)
4. Hamming Network And MAXNET
5. Backpropagation Algorithms.
6. Quickprop
7. Adaptive Resonance Theory (ART1)
8. Kohonen Self Organizing Map
9. Neocognitron

The Bidirectional associative memory does heteroassociative processing in which, association between pattern pairs is stored. Hopfield network is based on autoassociation. It associated the input pattern with the closest

stored pattern. ART1 and Kohonen network acts as classifier, which classifies a set of patterns, and the network recalls the information regarding class membership of the stored patterns when any of them is presented to the network later. Hamming network does classification based on the hamming distance between the input and stored patterns. Neocognitron applies concept of competitive learning.

1.3 Problem Statement

The aim of the thesis is that neural network has demonstrated its capability for solving complex pattern recognition problems. Commonly solved problems of pattern have limited scope. Single neural network architecture can recognize only few patterns. Relative performance of various neural network algorithms has not been reported in the literature.

The thesis discusses on various neural network algorithms with their implementation details for solving pattern recognition problems. The relative performance evaluation of these algorithms has been carried out. The comparisons of algorithms have been performed based on following criteria:

- (1) Noise in weights
- (2) Noise in inputs
- (3) Loss of connections
- (4) Missing information and adding information.

1.4 Organization of The Thesis

The brief introduction of the neural network for pattern recognition is given in chapter 1. Chapter ^{2&3} contains the fundamental of neural networks. We have presented different detailed neural network algorithms used in the pattern recognition. In chapter 4, the performance of various algorithms under different criterion have been studied and compared.

The conclusions are given in chapter 5. This chapter includes the topic for future in this field.

Chapter 2: Fundamentals Of Artificial Neural Networks

Chapter 2: Fundamentals of Artificial Neural Networks

Artificial neural networks have developed in wide variety of configurations. Despite of this apparent diversity, network paradigm has a great deal in common. In this chapter, recurring themes are briefly identified and discussed so they will be familiar. There are no of different answers possible to the question of how to define neural network. At one extreme, answer could be that neural networks are simply a class of mathematical algorithms, since a network can be regarded essentially as a graphic notation for a large class of algorithms. Such algorithms produce solutions to a no of specific problems. At the other end, the reply may be that these are synthetic networks that emulate the biological neural networks found in living organisms. In light of today's limited knowledge of biological neural networks and organisms, the more possible answer seems to be closer to the algorithmic one.

2.1 Biological Neurons and Their Artificial Models

A human brain consists of approximately 10^{11} computing elements called neurons. They communicate through a connection network of axons and synapses having a density of approximately 10^4 synapses per neuron. Our hypothesis regarding the modeling of the natural nervous system is that neurons communicate with each other by means of electrical impulses [37-39]. The neurons operate in a chemical environment that is even more important in terms of actual brain behavior. We thus can consider the brain

to be a densely connected electrical switching network conditioned largely by the biochemical processes. The vast neural network has an elaborate structure with very complex inter connections. The input to the network is provided by sensory receptors. Receptors deliver stimuli both from within the body, as well as from sense organs when the stimuli originate in the external world. The stimuli are in the form of electrical impulses that convey the information in the network of neurons. As a result of information processing in the central nervous system, the effectors are controlled and give human responses in the form of diverse actions.

We thus have a three stage systems, consisting of receptors, neural network, and effectors, in control of the organism and its actions.

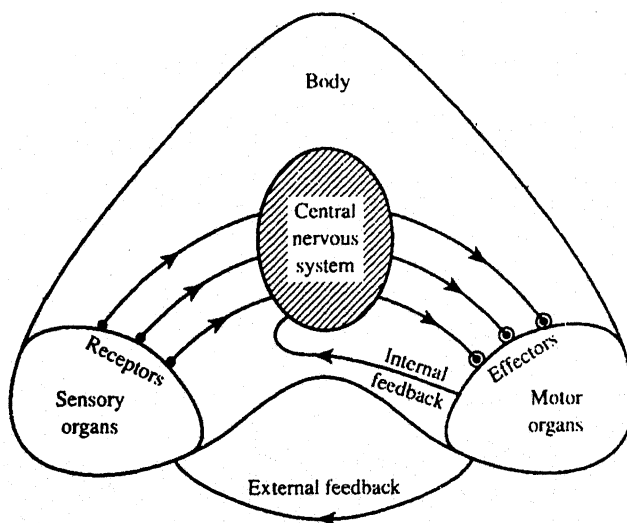


Fig: 2.1 Information Flow in Nervous System

A lucid, although rather approximate idea, about the information links in the nervous system is shown in fig2.1.

As we can see from the fig., the information is processed, evaluated, and compared with the stored information in the central nervous system. When necessary, commands are generated there and transmitted to the motor organs. Notice that motor organs are monitored in the central nervous system by feedback links that verify their actions. Both internal and external feedbacks control the implementation of commands.

2.1.1 Biological Neuron

The elementary nerve cell, called a neuron, is the fundamental building block of the biological neural network. Its schematic diagram has shown in fig 2.2. A typical cell has three major regions: the cell body, which is also called the soma, and axon, and the dendrites. Dendrites form a dendrite tree, which is a very fine bush of thin fibers around the neuron's body. Dendrites receive the information from neurons through axons-long fibers that serve as transmission lines. An axon is a long cylindrical connection that carries impulses from the neuron. The end part of an axon splits in to a fine arborization. Each branch of it terminates in a small end bulb all most touching the dendrites of neighboring neurons. The axon- dendrite contact organ is called a synapse. The synapse is where the neuron introduces its signal to neighboring neuron. The signal reaching a synapse and received by dendrites are electrical impulses.

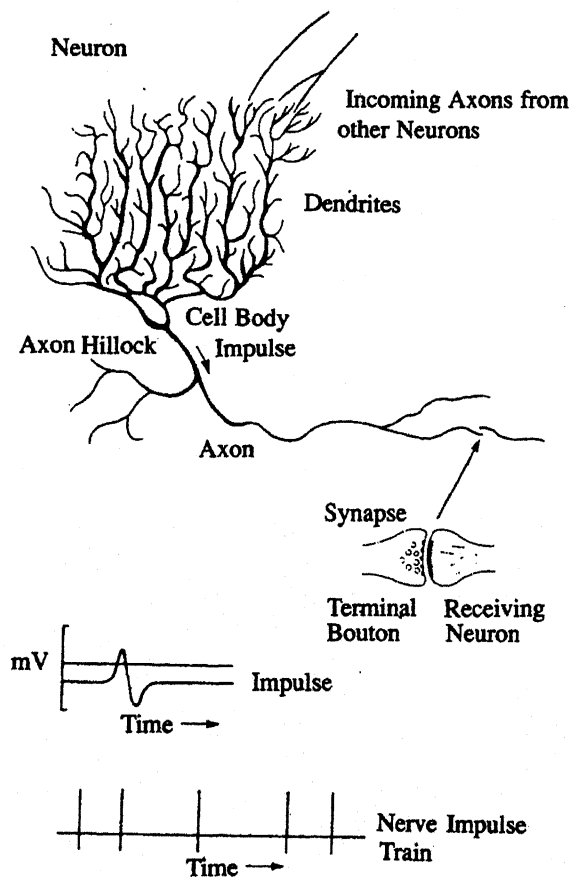


Fig: 2.2 Schematic Diagrams of a Neuron and a Sample of Pulse Train

2.1.2 Artificial Neuron

The artificial neuron was designed to mimic the first-order characteristics of the biological neuron. In essence, a set of inputs is applied, each representing the output of another neuron. Each input is multiplied by a corresponding weight, analogous to a

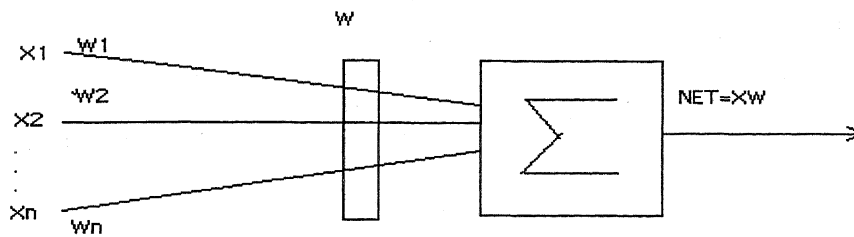


Fig. 2.3 Artificial Neuron

Synaptic strength and all of the weighted inputs are then summed to determine the activation level of the neuron. Fig. 2.3 shows a model that implements this idea. Despite the diversity of network paradigm, nearly all are based upon this configuration. Here, a set input leveled $x_1, x_2 \dots x_n$ is applied to artificial neuron. These inputs collectively referred to as the vector X , correspond to the signal in to the synapses of a biological neuron. Each signal is multiplied by an associated weight $w_1, w_2 \dots w_n$, before it is applied to the summation block, labeled Σ . Each weight corresponds to the “Strength” of a single biological synaptic connection.

The summation block, corresponding roughly to the biological cell body, adds all of the weighted inputs algebraically, producing an output that we call NET. This may be compactly stated in vector notation as follows,

$$NET = XW$$

Activation Function

The NET signal is usually further processed by an activation function F to produce the neuron's output signal, OUT. This may be simple linear function,

$$\text{OUT} = K (\text{NET})$$

Where K is a constant, a threshold function,

$$\text{OUT}=1 \quad \text{if } \text{NET} > T$$

$$\text{OUT}=0 \quad \text{otherwise}$$

Where T is constant threshold value, or a function that more accurately simulates the nonlinear transfer characteristic of the biological neuron and permits more general network functions.

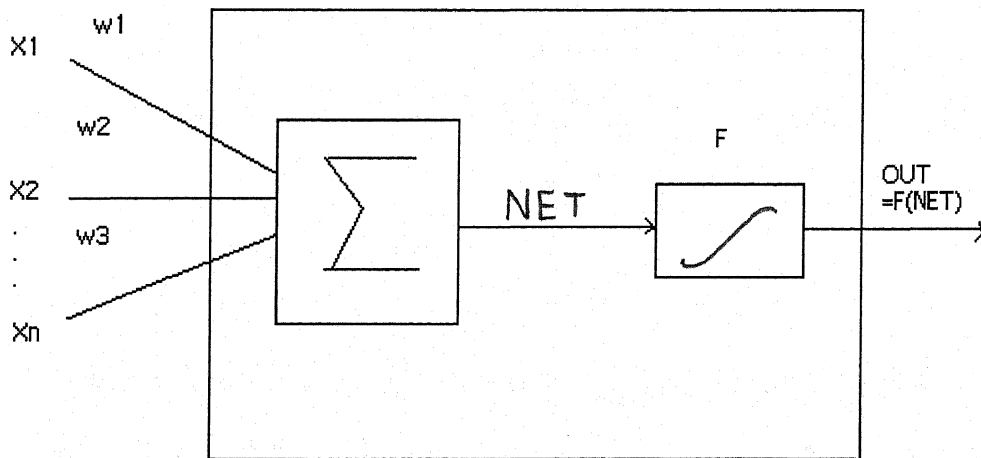


Fig. 2.4: Artificial Neuron with Activation Function

In fig. 2.4 the block labeled F accepts the NET output and produces the signal labeled OUT. If the F processing block compresses range of NET, so

that OUT never exceeds some low limits regardless of the values of NET, F is called a *squashing function*. The squashing function is often chosen to be the logistic function or “Sigmoid” meaning (S-shaped). This function is expressed mathematically as $F(x) = 1/(1 + e^{-x})$,

Thus

$$OUT = 1/(1 + e^{-NET}) = F(\overline{NET})$$

By analogy to analog in electronic systems, we may think of the activation function as defining a nonlinear gain for the artificial neuron. This gain is calculated by finding the ratio of the change in OUT to a small change in NET. Thus, gain is the slope of the curve as a specific excitation label.

Small input signal requires high gain through the network if they are to produce usable output; however, a large number of cascaded high- gain stages can saturate the output with the amplified noise (random variations) that is present in any realizable network. Also, large input signal will saturate high- gain stages, again eliminating any usable output. The central high-gain region of the logistic function solves the problem of processing small signals, while its regions of decreasing gain at positive and negative extremes are appropriate for large excitations.

Another commonly used activation function is the hyperbolic tangent. It is similar in shape to the logistic function and is often used by biologists as a mathematical model of nerve cell activation. Used as an artificial neural network activation function it is expressed as follows:

$$OUT = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

2.1.3 Single Layer Artificial Neural Networks

Although a single neuron can perform a certain simple pattern detection functions, the power of neural computation comes from connecting neurons in to networks. The simplest network is a group of neurons arranged in a layer as shown in the fig 2.5. Note that the circular notes on the left serve only to distribute the inputs: they perform no computation and hence will not be considered to constitute a layer. For this reason, they are shown as circles to distinguish them from the computing neurons, which are shown as squares. The set of inputs X has each of its elements connected to each artificial neuron through a separate weight. Early artificial neural networks were no more complex than this. Each neuron simply output a weighted sum of the inputs to the network. Actual artificial and biological networks may have many of the connections deleted, but full connectivity is shown for reasons of generality.

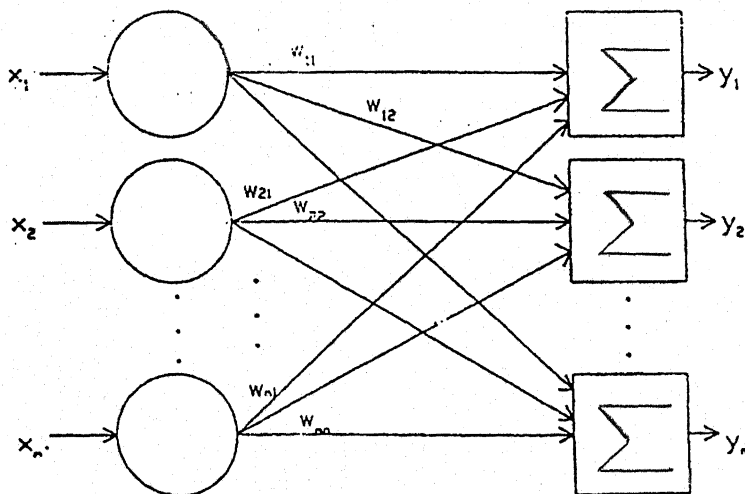


Fig. 2.5: Single Layer Neural Network

It is convenient to consider the weight to be elements of a matrix W . The dimensions of the matrix are m rows by n columns, where m is the no. of inputs and n the no. of neurons.

2.1.4 Multi Layer Artificial Neural Networks

Multilayer networks may be formed by simply cascading a group of single layers; the output of one layer provides the input to the subsequent layer.

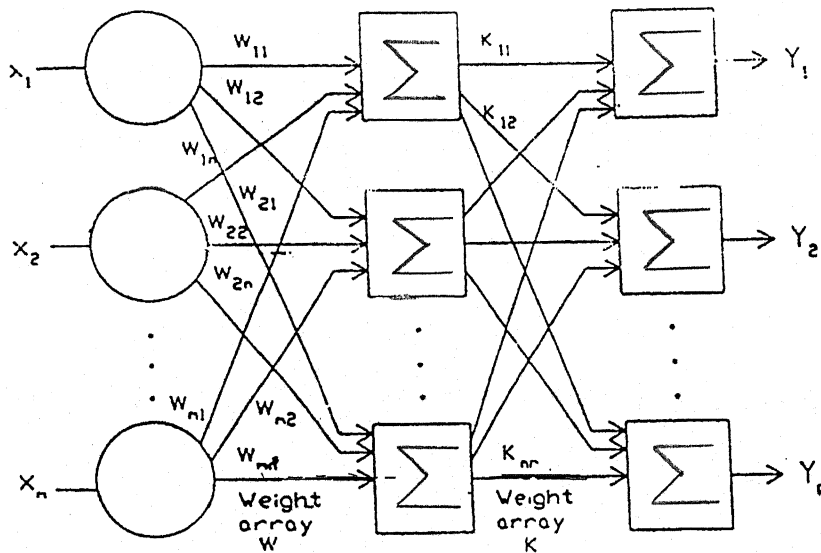


Fig 2.6: Two layer neural network

Non-linear Activation Function

Multilayer networks provide no increase in computational power over a single layer network unless there is a nonlinear activation function between layers. Calculating the output of a layer consists of multiplying the input vector by the first weight matrix, and then (if there is no nonlinear activation function) multiplying the resulting vector by the second weight matrix. This may be expressed as $(XW_1)W_2$. Since matrix multiplication is associative, the term may be regrouped and written: $X(W_1W_2)$

This shows that a two layer linear network is exactly equivalent to a single layer having a weight matrix equal to the product of the two weight matrices. Hence, any multilayer linear network can be replaced by an equivalent one-layer network. We point out that single layer networks are severely limited in their computational capability; thus, the non-linear activation functions are vital to the expansion of the network's capability beyond that of the single layer network.

Recurrent networks

The networks considered up to this point have no feedback connections, that is, connections through weights extending from the outputs of a layer to the inputs of the same or previous layers. This special class called non-recurrent or feed forward networks.

2.2 Terminologies, Notation, and Representation of Artificial Neural Networks

Many authors avoid the term “neuron” when referring to the artificial neuron, recognizing that it is a crude approximation of its biological model. The terms “neuron”, “cell”, and “unit” interchangeably as shorthand for “artificial neuron”, as these words are self-explanatory.

Learning algorithms, like artificial neural networks in general can be presented in either differential –equation or difference- equation form. The differential equation representation assumes that the processes are continuous, operating much like a large analog network. Viewing the biological system at a microscopic level, this is not true; the activation level of a biological neuron is determined by the average rate at which it emits discrete action potential pulses down its axons. This average rate is

commonly treated as an analog quantity but it is important to remember the underlined reality.

If one wishes to simulate artificial neural networks on an analog computer, differential equation representations are highly desirable. However, most work today is being done on digital computers, making the difference equation form most appropriate, as these equations can be converted easily in to computer programs. For this reason, the difference equation representation is used very much.

2.3 Training of Artificial Neural Networks

Of all of the interesting characteristics of artificial neural networks, none captures the imaginations like their ability to learn. Their training shows so many parallel to the intellectual development of human beings that it may seem that we have achieved a fundamental understanding of this process.

Objective of Training

A network is trained so that application of a set of inputs produces the desired set of outputs. Each such input set is referred to as a vector. Training is accomplished by sequentially applying input vectors, while adjusting networks weights according to a predetermined procedure. During training, the networks weights gradually converge to values such that each input vector produces the desired output vector.

Supervised Training

Training algorithms are categorized as supervised and unsupervised. Supervised training requires the pairing of each input vector with a target vector representing the desired output; together these are called a training pair. Usually a network is trained over a no. of such training pairs. An input vector is applied, the output of the network is calculated and compared to the corresponding target vector, and the difference (error) is fed back through

the network and weights are changed according to an algorithm that tends to minimize the error.

The vectors of the training set are applied sequentially, and errors are calculated and weights adjusted for each vector, until the error for the entire training set is at an acceptably low level

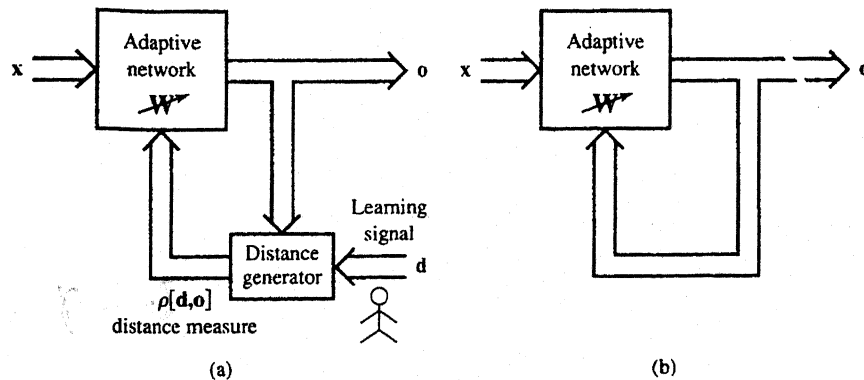


Fig: 2.7 Block Diagram for Explanation of Basic Learning Modes:

(A) Supervised Training (B) Unsupervised Training

In supervised learning we assume that at each instant of time when the input is applied, the desired response d of the system is provided by the teacher. This is illustrated in fig(2.7a). The distance $p[d, o]$ between the actual and the desired response serves as an error measure and is used to correct network parameters externally. Since we assume adjustable weights, the teacher may implement a reward-and –punishment scheme to adapt the network’s weight matrix W . For instance, in learning classifications of input patterns or situations with known responses, the error can be used to modify weights so that the error decreases. This mode of learning is very pervasive. Also, it is

used in many situations of natural learning. A set of input and output patterns called a training set is required for this learning mode.

Typically, supervised learning rewards accurate classification or associations and punishes those which yield inaccurate responses. The teacher estimates the negative error gradient direction and reduces the error accordingly. In many situations, the inputs, outputs and the computed gradient are deterministic, however, the minimization of error proceed over all its random realizations. As a result, most supervised learning algorithms reduce to stochastic minimization of error in multi-dimensional weight space.

Figure 2.7(b) shoes the block diagram of unsupervised learning. In learning without supervision, the desired response is not known; thus, explicit error information cannot be used to improve network behavior. Since no information is available as to correctness or incorrectness of responses, learning must somehow be accomplished based on observations of responses to inputs that we have marginal or no knowledge about.

Unsupervised Training

Despite many application successes, supervised training has been criticized as being biologically implausible; it is difficult to conceive of a training mechanism in the brain that compares desired and actual outputs, feeding processed corrections back through the network. If this were the brain's mechanism, where do the desired out puts patterns come from? How could the brain of an infant accomplish the self-organization that has been proven to exist in early development? Unsupervised training is a far more plausible model of learning in the biological system. Developed by Kohonen (1984)[40] and many others, it requires no target vectors for the outputs, and

hence, no comparisons to predetermined ideal responses. The training set consists solely of input vectors. The training algorithms modifies network weights to produce output vectors that are consistent; that is, both applications of one of the training vectors or applications of a vector that is sufficiently similar to it will produce the same patterns of outputs. The training process, therefore, expects the statistical properties of the training set and group's similar vector in to classes. Applying a vector from a given class to the input will produce a specific output vector, but there is no way to determine prior to training which specific output pattern will be produced by a given input vector class. Hence, the outputs of such a network must generally be transformed into a comprehensible form subsequent to the training process. This does not represent a serious problem. It is usually a simple matter to identify the input-output relationship established by the network.

Training Algorithms

Most of today's training algorithms have evolved from the concepts of D.O. Hebb [1961] [41]. He proposed a model for unsupervised learning in which the synaptic strength (weight) was increased if both the source and destination neuron were activated. In this way, often- used paths in the networks are strengthened, and the phenomena of habit and learning through repetition are explained.

An artificial neural network using hebbian learning will increase its network weights according to the product of the excitation levels of the source and destination neurons. In symbols:

$$w_{ij}(n+1) = w_{ij}(n) + \alpha OUT_i OUT_j$$

Where $w_{ij}(n)$ = the value of a weight from neuron i to neuron j prior to adjustment.

$w_{ij}(n+1)$ = the value of a weight from neuron i to neuron j after adjustment.

α = the learning- rate coefficient

OUT_i = The output of neuron i and input to neuron j

OUT_j = The output of neuron j .

Chapter 3: Neural Network Algorithms

Chapter 3: Neural Network Algorithms

The pattern recognition is a challenging problem. This problem has been attempted by using neural network algorithms. The associated difficulties for obtaining solutions have also been cited. To the best of our knowledge no attempt has been made to determine the capacity of various neural networks for recognition of all English alphabets A-Z by any single neural network algorithms. This chapter deals with details of existing techniques for pattern recognition and an attempt has been made to demonstrate the capacity of each algorithm.

3.1 Bidirectional Associative Memory (BAM)

The Bidirectional associative memory is heteroassociative, content-addressable memory. A BAM consists of neurons arranged in two layers say A and B. The neurons are bipolar binary. The neurons in one layer are fully interconnected to the neurons in the second layer. There is no interconnection among neurons in the same layer. The weight from layer A to layer B is same as the weights from layer B to layer A. This type of interconnection is called logical symmetry of interconnections [42]. It uses both forward and backward information flows to produce an associative search for stored patterns (Kosko 1987, 1988) [43-44]. Consider that stored in the memory are p vector association pairs known as,

$$\{(\mathbf{a}^{(1)}, \mathbf{b}^{(1)}), (\mathbf{a}^{(2)}, \mathbf{b}^{(2)}), \dots, (\mathbf{a}^{(p)}, \mathbf{b}^{(p)})\} \quad (3.1)$$

When the memory neurons are activated, the network evolves to a stable state of two-pattern reverberation, each pattern at output of one layer. The stable reverberation corresponds to a local energy minimum. The network's dynamics involves two layers of interaction. Because the memory process information in time and involves Bidirectional data flow, it differs in principle from a linear associator, although both networks are used to store association pairs. It also differs from the recurrent auto associative memory in its update mode.

3.1.1 Memory Architecture

The basic diagram of the Bidirectional associative memory is shown in fig. 3.1. Let us assume that an initializing vector \mathbf{b} is applied at the input to the layer A of neurons.

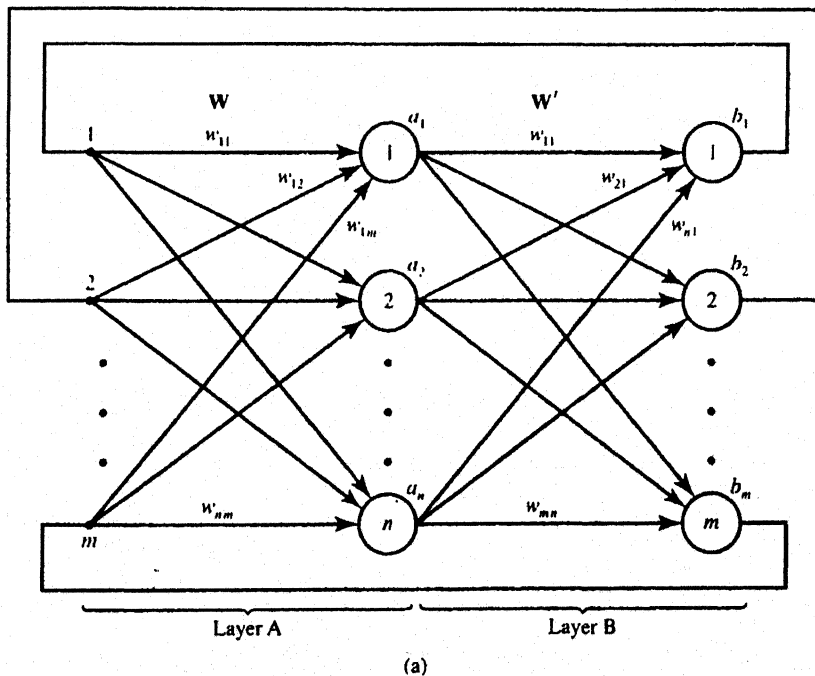


Fig.3.1: Bidirectional Associative Memory: General Diagram

The neurons are assumed to be bipolar binary. The input is processed through the linear connection layer and than through the bipolar threshold function as follows,

$$\mathbf{a}' = \Gamma[\mathbf{W}\mathbf{b}] \quad (3.1a)$$

Where $\Gamma[\cdot]$ is a nonlinear operator defined as,

$$\Gamma[\cdot] = \begin{bmatrix} \text{sgn}(\cdot) & 0 & \dots & 0 \\ 0 & \text{sgn}(\cdot) & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \text{sgn}(\cdot) \end{bmatrix}$$

This pass consists of matrix multiplication and a bipolar thresholding operation so that the i' th output is

$$a_i' = \text{sgn} \left(\sum_{j=1}^m w_{ij} b_j \right), \text{ for } i = 1, 2, \dots, n \quad (3.1b)$$

Assume that the thresholding as in (3.1a) and (3.1b) is synchronous, and the vector \mathbf{a}' now feeds the layer B of neurons. It is now processed in layer B through similar matrix multiplication and bipolar thresholding but the processing now uses the transposed matrix \mathbf{w}' of the layer B:

$$\mathbf{b}' = \Gamma[\mathbf{W}'\mathbf{a}'] \quad (3.1c)$$

Or for the j' th output we have

$$b_j' = \text{sgn} \left(\sum_{i=1}^n w_{ij} a_i' \right), \text{ for } j = 1, 2, \dots, m \quad (3.1d)$$

From now on the sequence of retrieval repeats as in (3.1a) or (3.1b) to compute \mathbf{a}'' , then as in (3.1c) or (3.1d) to compute \mathbf{b}'' , etc. the process continues until further updates of \mathbf{a} and \mathbf{b} stop. It can be seen that in terms of recursive update mechanism, the retrieval consists of the following steps

$$\text{First Forward Pass:} \quad \mathbf{a}^1 = \Gamma[\mathbf{W}\mathbf{b}^0]$$

$$\text{First Backward Pass:} \quad \mathbf{b}^2 = \Gamma[\mathbf{W}'\mathbf{a}^1]$$

Second Forward Pass: $\mathbf{a}^3 = \Gamma[\mathbf{W}\mathbf{b}^2]$ (3.2)

·
·
·

$k/2$ 'th Backward pass: $\mathbf{b}^k = \Gamma[\mathbf{W}'\mathbf{a}^{k-1}]$

Ideally, this back-and-forth flow of updated data quickly equilibrates usually in one of the fixed pairs $(\mathbf{a}^{(i)}, \mathbf{b}^{(i)})$ from (3.1). Let us consider in more detail the design of the memory that would achieve this aim.

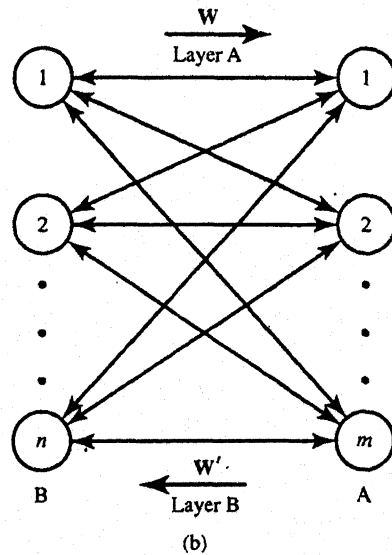


Fig 3.2: Bidirectional Associative Memory: Simplified Diagram

Fig (3.2) shows the simplified diagram of the Bidirectional associative memory often encountered in the literature. Layer A and B operate in an alternate fashion- first transferring the neuron's output signals towards the right by using matrix \mathbf{W} , and then toward the left by using the matrix \mathbf{W}' , respectively.

The Bidirectional associative memory maps bipolar binary vectors $\mathbf{a} = [a_1 a_2 \dots a_n]^t$, $a_i = \pm 1$, $i = 1, 2, \dots, n$, into vectors $\mathbf{b} = [b_1 b_2 \dots b_m]^t$, $b_i = \pm 1$, $i = 1, 2, \dots, m$, or vice versa. The mapping by the memory can also be performed for unipolar binary vectors. The input-output transformation is highly nonlinear due to the threshold – based state transitions.

For proper memory operation, the assumption needs to be made that no state changes are occurring in neurons of layers A and B at the same time. The data between layers must flow in a circular fashion: $A \rightarrow B \rightarrow A$, etc. The convergence of the memory is proved by showing that either synchronous or asynchronous state changes of a layer decrease the energy. The energy value is reduced during a single update. Because the energy of the memory is bounded from below, it will gravitate to fixed points. Since the stability of this type of memory is not affected by an asynchronous versus synchronous state update, it seems wise to assume synchronous operation. This will result in larger energy changes and, thus, will produce much faster convergence than asynchronous updates which are serial by nature and thus slow.

3.1.2 Association Encoding and Decoding

The coding of information (3.1) in to the Bidirectional associative memory is done using the customary outer product rule, or by adding p cross – correlation matrices. The formula for weight matrix is

$$W = \sum_{i=1}^p \mathbf{a}^{(i)} \mathbf{b}^{(i)t}$$

Where $\mathbf{a}^{(i)}$ and $\mathbf{b}^{(i)}$ are bipolar binary vectors, which are members of the i^{th} pair.

3.1.3 Algorithm

Step 1: The associations between pattern pairs are stored in the memory in the form of bipolar binary vectors with entries -1 and 1.

$$\{(\mathbf{a}^{(1)}, \mathbf{b}^{(1)}), (\mathbf{a}^{(2)}, \mathbf{b}^{(2)}), \dots, (\mathbf{a}^{(p)}, \mathbf{b}^{(p)})\}$$

Vector \mathbf{a} store pattern and is n-dimensional, \mathbf{b} is m-dimensional which stores associated output.

Step 2: Weights are calculated by

$$W = \sum_{i=1}^p \mathbf{a}^{(i)} \mathbf{b}^{(i)T}$$

Step 3: Test vector pair \mathbf{a} and \mathbf{b} is given as input.

Step 4: In the forward pass, \mathbf{b} is given as input and \mathbf{a} is calculated as

$$\mathbf{a} = \Gamma[\mathbf{W}\mathbf{b}]$$

Each element of vector \mathbf{a} is given by

$$a_i' = \text{sgn}\left(\sum_{j=1}^m w_{ij} b_j\right), \quad \text{for } i = 1, 2, \dots, n$$

Step 5: Vector \mathbf{a} is now given as input to the second layer during backward pass. Output of this layer is given by

$$\mathbf{b}' = \Gamma[\mathbf{W}\mathbf{a}]$$

Each element of vector \mathbf{b} is given by

$$b_j' = \text{sgn}\left(\sum_{i=1}^n w_{ij} a_i'\right), \quad \text{for } j = 1, 2, \dots, m$$

Step 6: If there is no further update then the process stops. Otherwise step 4 and 5 are repeated.

3.1.4 Storage Capacity

Kosko (1988)[44] has shown that the upper limit on the no. p of pattern pairs which can be stored and successfully retrieved is $\min(m, n)$. The substantiation for this estimate is rather heuristic. The memory storage capacity of BAM is

$$P \leq \min(m, n)$$

A more conservative heuristic capacity measure is used in a literature,

$$p = \sqrt{\min(m,n)}$$

3.1.5 Results

The network has 49 neurons in first layer and five neurons in the second layer. With this configuration the network is capable of storing six alphabets. The alphabets have been represented as a pattern grid of size 7×7 . From the pattern grid each alphabet is obtained

Character	Input	Output
A	a = [-1-1111-1-1-11-1-1-11-1-1-1-1-1-111111111-1-1-1-1-111-1-1-1-111-1-1-1-11] b = [-1-1-1-11]	-1-1-1-11
I	a = [-111111-1-1-1-11-1-1-1-1-1-11-1-1-1-1-1-11-1-1-1-111111-1] b = [-11-1-11]	-11-1-11
L	a = [1-1-1-1-1-1-11-1-1-1-1-1-11-1-1-1-1-1-11-1-1-1-1-1-11-1-1-1-1-1-1111111] b = [-111-1-1]	-111-1-1
M	a = [1-1-1-1-1-1111-1-1-1111-11-11-111-1-11-1-111-1-1-1-1-1-111-1-1-1-11] b = [-111-11]	-111-11
P	a = [11111-1-11-1-1-1-11-11-1-1-1-111-1-1-1-11-111111-1-11-1-1-1-1-11-1-1-1-1-1] b = [1-1-1-1-1]	1-1-1-1-1
X	a = [1-1-1-1-1-11-11-1-1-11-1-1-11-11-1-1-1-1-11-1-1-1-1-11-11-1-1-11-11-1-1-1-11] b = [11-1-1-1]	11-1-1-1

Table 3.1: Given input and Obtained output in BAM

in the form of a vector with entries 1 and -1. All the alphabets in the form vector **a** and its associated output as vector **b** is stored. Weight matrix is obtained by taking inner product of **a** and **b**. The network has been tested for A, I, L, M, X, and P. The network gives the correct associated output when these characters are given at the time of testing. The result has been shown in table 3.1.

3.1.6 Merits and Demerits

In BAM, a distorted input pattern may also cause correct heteroassociation at the output. The logical symmetry of interconnection severely hampers the efficiency of BAM in pattern storage and recall capability. It also limits their use for knowledge representation and inference [45]. There is limitation on number of pattern pairs, which can be stored and successfully retrieved

3.2 Hopfield Autoassociative Memory

The Hopfield network is an associative memory. The primary function of which is to retrieve in a pattern stored in memory, when an incomplete or noisy version of that pattern is presented.

In Hopfield model, each neuron has two states. The on state of neuron is denoted by the output +1 the off state is represented by -1. A pair of neurons i and j in the network are connected by weight w_{ij} , which denotes the contribution of output signal of neuron i to the potential acting on neuron j . There are two phases to the operation of the Hopfield network (i) storage phase and (ii) Retrieval phase. In the storage phase N- dimensional patterns are stored in the memory. These patterns can be retrieved during retrieval phase when these patterns are presented to the network as test vectors.

3.2.1 Algorithm

Step 1: n-dimensional vectors $a_1, a_2, a_3 \dots a_p$ are stored in the memory with entry +1 and -1.

Step 2: Weights of the network is calculated as

$$w_{ji} = \frac{1}{n} \sum_{\mu=1}^p a_{\mu j} a_{\mu i} \quad j \neq i$$
$$0 \quad j = i$$

The weights are not updated during iterations.

Step 3: n-dimensional probe vector x is presented to the network. The algorithm is initialized by,

$$a_j(0) = x_j \quad \text{for } j=1, 2, 3, \dots, n$$

Step 4: The elements of the vector $a(k)$ is updated by

$$a_j(k+1) = \text{sgn} \left[\sum_{i=1}^n w_{ji} a_i(k) \right]$$

Step 5: During any iteration if

$$a_j(k+1) = a_j(k)$$

Then there is no further iteration that is vector a becomes stable. Otherwise step 4 is repeated.

Step 6: The stable state a_{fixed} is the output vector of the network.

$$Y = a_{\text{fixed}}$$

3.2.2 Storage Capacity

The storage capacity of the Hopfield network is given by

$$P_{\max} = \frac{n}{2 \ln 2}$$

Where P_{\max} is the maximum numbers of fundamental memory that can be stored. n is the dimension of vector p .

3.2.3 Result

The network has two layers. There are 49 neurons in each layer. 7×7 pattern grid is used to represent a character. With this configuration the network is able to store and successfully recall six characters. Six characters Y, O, T, L, M, and I are stored in the network. Each is represented as 49-dimensional vectors with entry +1 and -1. Weight matrix is found from the stored patterns. After storage, when these characters are presented individually to the network they are recognized correctly. This has been shown in table 3.2. When A, I, L, M, X and P are stored, all these are not recognized correctly at the time of testing. I, L, M, X and P recognized correctly. When character A is presented as probe vector, the network gives output as shown in Table 3.3.

3.2.4 Merits and Demerits

Hopfield network recognizes well even when an input vector that is equal to the stored patterns plus random error is presented. The Hopfield Network demonstrates the power of recurrent neural processing within a parallel architecture. The recurrences through the thresholding layer of neurons eliminated the noise added to the initializing input vector [46-47].

Hopfield associative memory has capacity limitations. Capacity limitation causes convergence to spurious states. Spurious states represent stable states of the network that are different from the stored pattern or fundamental memories of the network [48]. It also causes difficulty with recovery of stored patterns if the patterns are closed to each other in the hamming distance

Character	Input	Output
I	<pre> o * * * * * o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o * * * * * o </pre>	<pre> o * * * * * o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o * * * * * o </pre>
L	<pre> * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * * * * * </pre>	<pre> * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * * * * * </pre>
M	<pre> * o o o o o * * * o o o * * * o * o * o * * o o * o o * * o o o o o * * o o o o o * * o o o o o * </pre>	<pre> * o o o o o * * * o o o * * * o * o * o * * o o * o o * * o o o o o * * o o o o o * * o o o o o * </pre>
O	<pre> o * * * * * o * o o o o o * * o o o o o * * o o o o o * * o o o o o * * o o o o o * o * * * * * o </pre>	<pre> o * * * * * o * o o o o o * * o o o o o * * o o o o o * * o o o o o * * o o o o o * o * * * * * o </pre>
T	<pre> * * * * * o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o </pre>	<pre> * * * * * o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o </pre>
Y	<pre> * o o o o o * o * o o o * o o o * o * o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o </pre>	<pre> * o o o o o * o * o o o * o o o * o * o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o </pre>

Table 3.2: Given Input and Obtained Output in Hopfield Network

Character	Input	Output
A	<pre> o o * * * o o o * o o o * o * o o o o o * * * * * * * o o o o o * * o o o o o * * o o o o o * </pre>	<pre> * o o o o o * * * o o o * o * o o o o o * * o o * o o * * o o o o o * * o o o o o * * o o o o o * * o o o o o * </pre>
I	<pre> o * * * * * o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o * * * * * o </pre>	<pre> o * * * * * o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o * * * * * o </pre>
L	<pre> * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * * * * * </pre>	<pre> * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * o o o o o o * * * * * </pre>
M	<pre> * o o o o o * * * o o o * * * o * o * o * * o o * o o * * o o o o o * * o o o o o * * o o o o o * </pre>	<pre> * o o o o o * * * o o o * * * o * o * o * * o o * o o * * o o o o o * * o o o o o * * o o o o o * </pre>
P	<pre> * * * * * o * o o o o * * o o o o o * * o o o o o * * * * * * o * o o o o o * o o o o o </pre>	<pre> * * * * * o * o o o o * * o o o o o * * o o o o o * * * * * * o * o o o o o * o o o o o </pre>
X	<pre> * o o o o o * o * o o o * o o o * o * o o o o o * o o o o o * o * o o o * o o o * o * o o o o o * </pre>	<pre> * o o o o o * o * o o o * o o o * o * o o o o o * o o o o o * o * o o o * o o o * o * o o o o o * </pre>

Table 3.3: Another set of Input and Obtained Output in Hopfield Network

3.3 Feature Recognition Neural Network

Feature recognition neural network is also used for character recognition. This network learns the patterns by remembering their different segments. It uses recognition by parts technique by remembering the different sections of the pattern. Thus noise or deformation in one section of the pattern does not affect the overall recognition process. This is the basis of the development of the feature recognition algorithm [49].

This network has two levels. The first level detects the sub patterns. The second level is responsible for detecting the patterns themselves and provides the output class. The pattern grid distributes the inputs over the first level. The neurons in the first level detect the sub patterns and feed a single neuron in level two. That fires if all the sub patterns are detected. The neurons of second level fires whenever, any one of the former neuron detects the sub pattern.

To recognize any pattern the network goes to following steps:

- (1) The network learns all the training patterns and remembers their sub patterns.
- (2) The new pattern (test patterns) is also divided in to subpattern.
- (3) Each of the subpatterns is compared individually against the corresponding sub patterns of training patterns and suitable match is found.
- (4) Then it finds the total number of subpattern that matched for any pattern and finds the closest overall match.

3.3.1 Algorithm

Step 1: A set of training vectors

$$A_p = \{a_{1p}, a_{2p}, \dots, a_{np}\} \quad \text{for } p=1, 2, \dots, p$$

Is given as input where $a_{ip}=0$ or 1

n is the dimension of training vector.

Step 2: The weights are

$$W_p = \{w_{1p}, w_{2p}, \dots, w_{np}\}$$

Where $w_{ip} = 1$ if $a_{ip} = 1$

-1 if $a_{ip} = 0$

Step 3: Each pattern is divided in to s number of subpatterns which is given by

$$A_{sp} = \{a_{1sp}, a_{2sp}, \dots, a_{nsp}\} \text{ For } s = 1, 2, \dots, S$$

$$p = 1, 2, \dots, P$$

m is the dimension of vector obtained from each subpattern.

Step 4: The weight matrix formed from subpatterns is

$$W_{sp} = \{w_{1sp}, w_{2sp}, \dots, w_{msp}\}$$

Where $w_{jsp} = 1$ if $a_{jsp} = 1$

-1 if $a_{jsp} = 0$

For $j = 1, 2, \dots, m$

Step 5: For all the subpattern of every training patterns threshold is calculated as

$$\theta_{sp} = A_{sp}(W_{sp})' \quad \text{for } s=1, 2, \dots, S$$

$$p = 1, 2, \dots, P$$

Step 6: Test pattern is presented to the network and stored as the vector

$$at = (at_1, at_2, \dots, at_n)$$

Step 7: Test pattern is divided into subpatterns

$$ats_s = (ats_{s1}, ats_{s2}, \dots, ats_{sm}) \quad \text{for } s=1, 2, \dots, S$$

Step 8: Inner product of sub patterns of test patterns with sub patterns of every stored patterns is calculated and stored in the vector \mathbf{p} .

$$\mathbf{p}_{sp} = A_{sp}(\mathbf{ats})^t \text{ for } s=1,2,\dots,S$$

$$p=1,2,\dots,P$$

Step 9: Each threshold θ_{sp} is compared with p_{sp} and variable parfire is set accordingly.

$$\theta_{sp} = p_{sp} \text{ Partial firing } \text{parfire}_{sp} = 1$$

Otherwise $\text{parfire}_{sp} = 0$

Step 10: overall firing of each neuron of output layer is calculated as

$$\text{fire}_p = \sum_{s=1}^S \text{parfire}_{sp}$$

Step 11: Maximum of all fire_p for $p=1,2,\dots,P$ is selected, which gives the out class p .

3.3.2 Results

Three layers network configuration has been taken. The output layer has 26 neurons one corresponding to each character. 26 characters in the form of 9×9 pattern grids are given as training pattern to the network. Each pattern is divided into 9 sub patterns of size 3×3 . The network correctly classifies 23 characters when they are presented individually to the network as test pattern. The neuron corresponding to the character no. has maximum value of the variable fire . The network can not differentiate O and Q from C and also R from P. This has been shown in table 3.4.

3.3.3 Merits and Demerits

The network uses simple integer weights. It converges in a single iteration. It gives the results instantaneously without weighting for any stabilization period convergence is guaranteed.

The network is complex and it involves a large no. of neurons. Although its structure is simpler when compared to Neocognitron. But the no. of neurons required is greater than as compared to other methods. But this demerit is overshadowed from the point of view of storage capacity.

Character	Input Pattern	Input Sub Pattern	Output
A	○ ○ ○ * * * ○ ○ ○ ○ ○ * ○ ○ ○ * ○ ○ ○ * ○ ○ ○ ○ ○ * ○ * ○ ○ ○ ○ ○ ○ ○ * * * * * * * * * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ *	○ ○ ○ * * * ○ ○ ○ ○ ○ * ○ ○ ○ * ○ ○ ○ * ○ ○ ○ ○ ○ * ○ * ○ ○ ○ ○ ○ ○ ○ * * * * * * * ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ *	Maximu m fire is of neuron 1
B	* * * * * * * ○ * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * * * * * * ○ ○ * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * * * * * * * ○	* * * * * * * ○ * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * * * * * * ○ ○ * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * * * * * * * ○	Maximu m fire is of neuron 2
C	○ ○ * * * * ○ ○ ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ ○ * ○ ○ ○ ○ ○ ○ ○ ○ * ○ ○ ○ ○ ○ ○ ○ ○ * ○ ○ ○ ○ ○ ○ ○ * ○ * ○ ○ ○ ○ ○ * ○ ○ ○ * * * * ○ ○	○ ○ * * * * ○ ○ ○ * ○ ○ ○ ○ ○ * ○ * ○ ○ ○ ○ ○ ○ ○ * * ○ ○ ○ ○ ○ ○ ○ ○ * ○ ○ ○ ○ ○ ○ ○ ○ * ○ ○ ○ ○ ○ ○ ○ ○ * ○ ○ ○ ○ ○ ○ ○ * ○ * ○ ○ ○ ○ ○ * ○ ○ ○ * * * * ○ ○	Maximu m fire is of neuron 3

O	<div> <div>○ ○ *</div> <div>○ ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>○ *</div> <div>○ ○</div> </div>	<div> <div>○ ○ *</div> <div>○ *</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>○ *</div> <div>○ ○</div> </div>	Max. fire is of neurons 3 and 15
Q	<div> <div>○ ○ *</div> <div>○ ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>○ *</div> <div>○ ○</div> </div>	<div> <div>○ ○ *</div> <div>○ *</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>○ *</div> <div>○ ○</div> </div>	Max. fire is of neurons 15
P	<div> <div>* * *</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* * *</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> </div>	<div> <div>* * *</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* * *</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> <div>* ○ ○</div> </div>	Max. fire is of neuron 15

R	*	*	*	*	*	*	*	*	*	o	*	*	*	*	*	*	*	*	o		Max. fire
	*	o	o	o	o	o	o	o	o	*	*	o	o	o	o	o	o	o	*	is	of
	*	o	o	o	o	o	o	o	o	*	*	o	o	o	o	o	o	o	*	neurons	
	*	o	o	o	o	o	o	o	o	*	*	o	o	o	o	o	o	o	*	15	and
	*	*	*	*	*	*	*	*	*	o	*	*	*	*	*	*	*	*	o	17	
	*	o	o	*	o	o	o	o	o	o	*	o	o	*	o	o	o	o	o		
	*	o	o	o	*	o	o	o	o	o	*	o	o	o	*	o	o	o	o		
	*	o	o	o	o	*	o	o	o	o	*	o	o	o	*	o	o	o	o		
	*	o	o	o	o	o	o	*	o	o	*	o	o	o	*	o	o	o	o		
	*	o	o	o	o	o	o	*	o	o	*	o	o	o	*	o	o	o	o		

Table 3.4: Given Input and Obtained Output in FRNN

3.4 Hamming Network and MAXNET

This is a two-layer classifier of binary bipolar vectors. The first layer of hamming network itself is capable of selecting the stored class that is at minimum HD value to the test vector presented at the input. The second layer MAXNET only suppresses outputs other than the maximum output node of the first layer[50].

The hamming network is of the feed forward type. The number of output neurons in this part equals the number of classes. The strongest response of a neuron of this layer indicated the minimum HD value between the input vector and the class this neuron represents. The second layer is MAXNET, which operates as a recurrent network. It involves both excitatory and inhibitory connections. The block diagram of this network is shown in the fig 3.3; it is a minimum hamming distance classifier which selects the stored classes that are at a minimum HD value to the noisy or incomplete argument vector presented at the input.

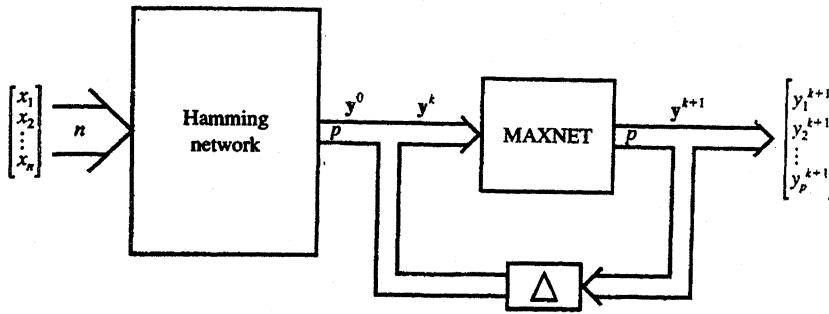


Fig.3.3: Block diagram of the minimum HD classifier

This selection is essentially performed solely by the Hamming network. The Hamming network is of the feed forward type and constitutes the first layer of the classifier. The p -class-hamming network has p output neurons. The strongest response of a neuron is indicative of the minimum HD value between the input and the category this neuron represents. The second layer of the classifier is called MAXNET and it operates as a recurrent recall network in an auxiliary mode. Its only function is to suppress values at MAXNET output nodes other than the initially maximum output node of the first layer. The proper part of the classifier is the Hamming network responsible for matching of the input vector with stored vector. The expanded diagram of the hamming network for classification of bipolar binary n -tuple input vectors are shown in fig 3.4. The purpose of the layer is to compute, in a feed forward manner, the values of $(n-HD)$, where HD is the hamming distance between the search argument and the encoded class prototype vector. Assume that the n -tuple prototype vector of the m th class is $s^{(m)}$, for $m=1,2,3,\dots,p$, and the n -tuple vector is x . Note that the entries of the weight vectors w_m defined as,

$$w_m = [w_{m1} w_{m2} \dots w_{mn}]^T, \text{ for } m=1, 2 \dots p$$

Connects inputs to the m th neuron, which performs as the class indicator.

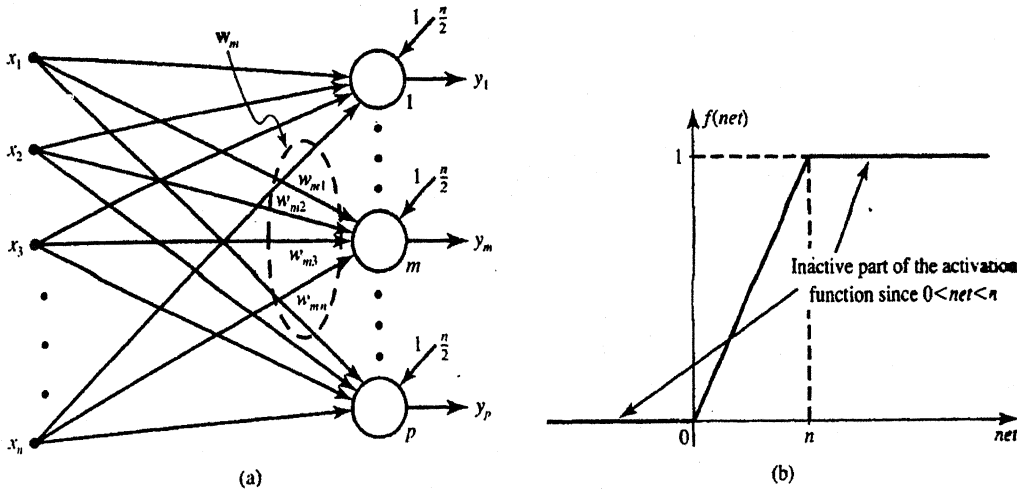


Fig. 3.4: Hamming Network for n-bit bipolar binary vectors representing p classes: (a) Classifier Network and (b) Neurons Activation Function

A vector classifier with p outputs, one for each class, can be conceived such that mth output is one if and only if $\mathbf{x} = \mathbf{s}^{(m)}$. This would require that the weights be $\mathbf{w}_{(m)} = \mathbf{s}^{(m)}$. The classifier outputs are $\mathbf{x}^t \mathbf{s}^{(1)}$, $\mathbf{x}^t \mathbf{s}^{(2)}$,, $\mathbf{x}^t \mathbf{s}^{(m)}$,, $\mathbf{x}^t \mathbf{s}^{(p)}$. When $\mathbf{x} = \mathbf{s}^{(m)}$, only the m'th output is n, provided the classes differ from each other, and assuming ± 1 entries of \mathbf{x} . The scalar product of the vectors has been used here as an obvious measure for vector matching.

The scalar product $\mathbf{x}^t \mathbf{s}^{(m)}$ of two bipolar binary n-tuple vectors can be written as the total number of positions in which the two vectors agree minus the number of positions in which they differ. Understandably, the number of positions in which two vectors agree is n-HD. The equality is written,

$$\mathbf{x}^t \mathbf{s}^{(m)} = (n - HD(\mathbf{x}, \mathbf{s}^{(m)})) - HD(\mathbf{x}, \mathbf{s}^{(m)}) \quad (3.4.1)$$

This is equivalent to

$$\frac{1}{2} \mathbf{x}^t \mathbf{s}^{(m)} = \frac{n}{2} - HD(\mathbf{x}, \mathbf{s}^{(m)}) \quad (3.4.2)$$

We can now see that the weight matrix \mathbf{W}_H of the hamming network can be created by encoding the class vector prototypes as rows in the form as below,

$$\mathbf{W}_H = \frac{1}{2} \begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \dots & s_n^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \dots & s_n^{(2)} \\ \vdots & \vdots & \dots & \vdots \\ s_1^{(p)} & s_2^{(p)} & \dots & s_n^{(p)} \end{bmatrix} \quad (3.4.3)$$

Where the $\frac{1}{2}$ factor is convenient for scaling purposes. Now the network with input vector \mathbf{x} yield the value of $\frac{1}{2} \mathbf{x}^t \mathbf{s}^{(m)}$ act the input to the node m for $m = 1, 2, \dots, p$. Adding the fixed bias value of $n/2$ to the input of each neuron results in the total input net_m .

$$net_m = \frac{1}{2} \mathbf{x}^t \mathbf{s}^{(m)} + \frac{n}{2}, \text{ for } m=1,2,\dots,p \quad (3.4.4)$$

Using the identity (3.4.2), net_m can be expressed as,

$$net_m = n - HD(\mathbf{x}, \mathbf{s}^{(m)}) \quad (3.4.5)$$

Let us apply neurons in the hamming network with activation function as in fig 3.4(b). The neurons need to perform only the linear scaling of (3.4.5) such that $f(net_m) = (1/n)net_m$, for $m=1,2,\dots,p$. Since inputs are between 0 and n , we obtain the outputs of each node scaled down to between 0 and 1. Further more, the number of the node with the highest output indeed indicated the class number to which \mathbf{x} is at the smallest HD. A perfect match of input vector to class m , which is equivalent to the condition $HD=0$, is signaled by $f(net_m)=1$. An input vector that is the compliment of the prototype of class m would result in $f(net_m)=0$. The response of the hamming network essentially terminated the classification in which only the first layer from fig. 3.3 computes the relevant the matching the score values.

As, seen, the classification by the hamming network is performed in a feed forward and instantaneous manner.

MAXNET needs to be employed as a second layer only for cases in which an enhancement of the initial dominant response of the m 'th node is required. As a result of MAXNET recurrent processing, the m 'th node responds positively, as opposed to all remaining nodes whose responses should have decayed to zero. As shown in fig. 3.5,

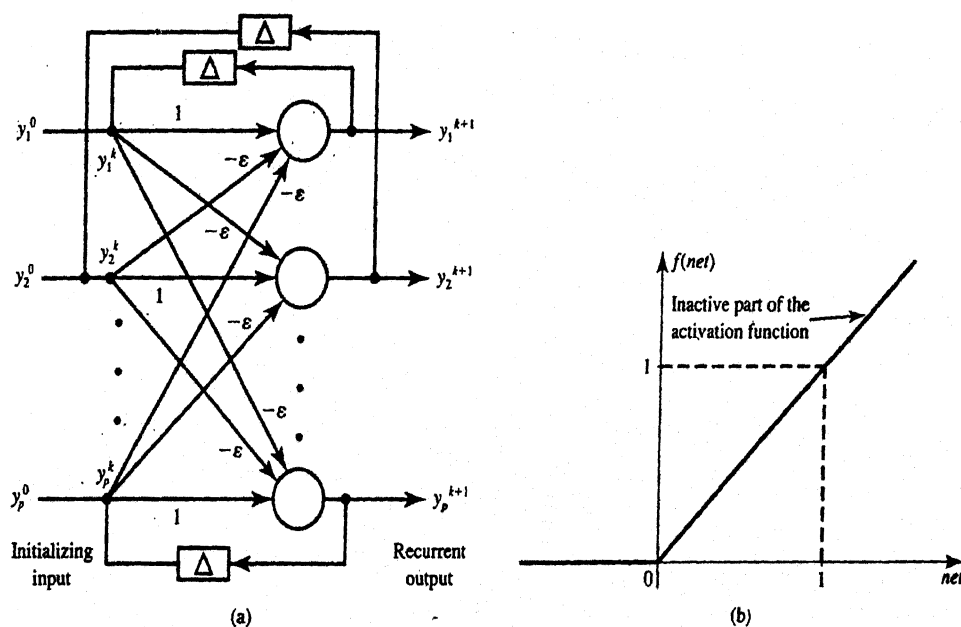


Fig. 3.5 MAXNET for p classes: (a) Network Architecture and (b) Neurons Activation Function

MAXNET is recurrent network involving both excitatory and inhibitory connection. The excitatory connection within the network is implemented in the form of a single positive self – feed back loop with a weighting coefficient of one. All the remaining connections of this fully coupled feed back network are inhibitory. They are represented as $M-1$ cross feed back synapses with coefficients $-\epsilon$ from each output. The second weight matrix W_M of size $p \times p$ is thus of the form,

$$\mathbf{W}_M = \begin{bmatrix} 1 & -\varepsilon & -\varepsilon & \dots & -\varepsilon \\ -\varepsilon & 1 & -\varepsilon & \dots & -\varepsilon \\ -\varepsilon & \dots & \dots & \dots & \\ \vdots & & \ddots & & \vdots \\ -\varepsilon & -\varepsilon & & -\varepsilon & 1 \end{bmatrix} \quad (3.4.6)$$

Where ε must be bounded $0 < \varepsilon < 1/p$. the quantity ε can be called the literal interaction coefficient. With the activation function as shown in fig. 3.5 and the initializing inputs fulfilling conditions,

$$0 \leq y_i^0 \leq 1, \text{ for } i=1, 2 \dots p$$

the MAXNET network gradually suppresses all but the largest initial network excitation. When initialized with the input vector \mathbf{y}^0 , the network starts processing it by adding positive self feed back and negative cross feed back. As a result of a number recurrence, the only unsuppressed node will be the one with the largest initializing entry y_m^0 . This means that the only nonzero out put response node is the node closest to the input vector argument in HD sense. The recurrent processing by MAXNET leading to this response is

$$\mathbf{y}^{k+1} = \Gamma[\mathbf{W}_M \mathbf{y}^k] \quad (3.4.7)$$

where Γ is a nonlinear diagonal matrix operator with entries $f(\cdot)$ given below:

$$f(net) = \begin{cases} 0, & net < 0 \\ net, & net \geq 0 \end{cases} \quad (3.4.8)$$

Each entry of the updated vector \mathbf{y}^{k+1} decreases at the k 'th recursion step of (3.4.7) under the MAXNET update algorithm, with the largest entry decreasing slowest. This is due to the conditions on the entries on the entries of matrix \mathbf{W}_M as in (3.4.6) specifically, due to the condition $0 < \varepsilon < 1/p$.

Assume that $y_m^0 > y_i^0, i = 1, 2, \dots, p$ and $i \neq m$. During the first recurrence, all entries of y^1 are computed on the linear portion of $f(net)$. The smallest of all y^0 entries will first reach the level $f(net)=0$, assumed at the k 'th step. The clipping of one output entry slows down the decrease of y_m^{k+1} in all forthcoming steps. Then, the second smallest entry of y^0 reaches $f(net)=0$. The process repeats it self until all values except for one, at the output of the m 'th node, remain at nonzero value.

3.4.1 Algorithm

Step 1: Consider that patterns to classified are $\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_p$, each pattern is n dimensional. The weights connecting inputs to the neuron of hamming network is given by weight matrix.

$$\mathbf{W}_H = \frac{1}{2} \begin{bmatrix} \mathbf{a}_{11} & \mathbf{a}_{12} & \dots & \mathbf{a}_{1n} \\ \mathbf{a}_{21} & \mathbf{a}_{22} & \dots & \mathbf{a}_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \mathbf{a}_{p1} & \mathbf{a}_{p2} & \dots & \mathbf{a}_{pn} \end{bmatrix} \quad (3.4.9)$$

Step2: n -dimensional input vector \mathbf{x} is presented to the input.

Step3: Net input of each neuron of hamming network is

$$net_m = \frac{1}{2} \mathbf{x}^t \mathbf{s}^{(m)} + \frac{n}{2}, \quad \text{for } m = 1, 2, \dots, p$$

Where $n/2$ is fixed bias applied to the input of each neuron of this layer.

Step 4: Output of each neuron of first layer is

$$f(net_m) = \frac{1}{n} net_m \quad \text{for } m = 1, 2, \dots, p$$

Step 5: Output of hamming network is applied as input to MAXNET

$$y^0 = f(net_m)$$

Step 6: Weights connecting neurons of hamming network and MAXNET is taken as

$$W_M = \begin{bmatrix} 1 & -\varepsilon & -\varepsilon & \dots & -\varepsilon \\ -\varepsilon & 1 & -\varepsilon & \dots & -\varepsilon \\ -\varepsilon & \dots & \dots & \dots & \dots \\ \vdots & & & \ddots & \vdots \\ -\varepsilon & -\varepsilon & & -\varepsilon & 1 \end{bmatrix}$$

Where the value of ε must be bounded $0 < \varepsilon < 1/p$. the quantity ε is called the lateral interaction coefficient. Dimension of W_M is $p \times p$

Step 7: The output of MAXNET is calculated as,

$$y^{k+1} = \Gamma[W_M y^k] \quad k=1, 2, 3, \dots \text{ denotes the no of}$$

iterations.

Where Γ is a nonlinear diagonal matrix operator with entries $f(\cdot)$ given as,

$$f(net) = \begin{cases} 0, & net < 0 \\ net, & net \geq 0 \end{cases}$$

Step 8: Each element of y^{k+1} decreases with increase of number of iterations (k). Maximum of y^{k+1} gives the class to which test vector x belongs.

3.4.2 Storage Capacity

The network has stringent capacity limits. The hamming network is capable of classifying all the stored patterns during testing.

3.4.3 Result

The network has 26 neurons in both layers. All the 26 characters are stored as 49 dimensional vectors whose entries are +1 and -1. +1 represents an on pixel and -1 represents an off one. The classifier correctly classifies each of 26 characters when these are presented individually to the network for testing. The value of lateral interaction coefficient used is 0.005. The output of neuron corresponding to the character number is highest.

3.4.4 Merits and Demerits

The network architecture is very simple. This network is a counter part of Hopfield auto associative network. The advantage of this network is that it involves less number of neurons and less number of connections in comparison to its counter part. There is no capacity limitation.

The hamming network retrieves only the closest class index and not the entire prototype vector. It is not able to restore any of the key patterns. It provides passive classification only. This network does not have any mechanism for data restoration.

3.5 Adaptive Resonance Theory 1 (ART1)

This network was developed by Carpenter and Grossberg [51,52] and is called an adaptive resonance theory 1 (ART1) network. It serves the purpose of cluster discovery. This network learns the clusters in an unsupervised mode. ART1 network can accommodate new clusters without affecting the storage or recall capabilities for clusters already learn.

The network produces the clusters by itself, it such clusters are identified in input data, and stores the clustering the information about patterns of features without a priori information about the possible no. and type of clusters. Essentially the network “follows the leader” after it originates the

first cluster with the first input pattern received. It then creates the second cluster if the distance of the second pattern exceeds a certain threshold, otherwise the pattern is clustered with first cluster. This process of pattern inspection followed by either new cluster origination or acceptance of pattern to the old cluster is the main step of the ART1 network production.

The central part of ART1 network computes the matching score reflecting the degree of similarity of the present input to the previously encoded clusters. The topmost layer of the network does this. This part of network is functionally identical to the hamming network and MAXNET. The initializing input to the m 'th node of MAXNET is the familiar scalar product similarity major between the input \mathbf{x} and the vector \mathbf{W}_m . We thus have the initial matching scores of values

$$y_m^0 = \mathbf{w}_m^t \mathbf{x} \quad \text{For } m=1, 2, \dots, M \quad (3.5.1)$$

$$\text{Where } \mathbf{w}_m = [\mathbf{w}_{1m}, \mathbf{w}_{2m}, \dots, \mathbf{w}_{nm}]^t.$$

Note that the double subscript convention for weights w_{ij} is not followed. The first weight index denotes input node number "from"; the second index denotes node number "to". This is to conform to common notation used in the technical literature for this network. The activation function $f(\text{net})$ for the MAXNET neuron is shown in fig (3.4a) and given by fig(3.4b). It is also assume that a unit delay element stores each MAXNET neuron output signal during the unity time Δ during recursions before it arrives back at the top layer node input. The input of the topmost layer is initialized with vector \mathbf{y}^0 , entries of which are computed as matching scores (3.5.1), and thereafter the layer undergoes the recurrent updates. We thus have for this portion of the network

$$\mathbf{y}^{k+1} = \Gamma[\mathbf{W}_M \mathbf{y}^k] \quad (3.5.2)$$

The initializing matching scores vector $y^0 = \text{net}^0$ for (3.5.2) and for the top layer recurrences is given by simple feed forward mapping,

$$y^0 = \mathbf{W}\mathbf{x} \quad (3.5.3)$$

Where \mathbf{W} is the bottom to top processing weight matrix containing entries w_{ij} as follows,

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{n1} \\ w_{12} & w_{22} & \cdots & w_{n2} \\ \vdots & \vdots & \cdots & \vdots \\ w_{1M} & w_{2M} & \cdots & w_{nM} \end{bmatrix}$$

As for the MAXNET network, the single nonzero output for a large enough recursion index k , y_j^{k+1} , is produced by the j 'th top layer neuron. For this winning neuron we have,

$$y_j^0 = \sum_{i=1}^n w_{ij} x_i = \max_{m=1,2,\dots,M} \left(\sum_{i=1}^n w_{im} x_i \right) \quad (3.5.4)$$

3.5.1 Algorithm

Step 1: Proper value of vigilance threshold ρ is set, and for the n -tuple input vectors and M top layer neurons the weights are initialized. The weight matrices \mathbf{W} , \mathbf{V} are of dimension $(M \times n)$ and each is initialized with identical entries.

$$\mathbf{W} = \begin{bmatrix} 1 \\ 1+n \end{bmatrix}$$

$$\mathbf{V} = [1]$$

$$0 < \rho < 1$$

Step 2: Binary Unipolar input vector \mathbf{x} is presented at input nodes,

$$x_i = 0, 1, \text{ for } i=1, 2, \dots, n$$

Step 3: Matching scores are computed for each patterns. The output of the forward network is calculated as,

$$y_m^0 = \sum_{i=1}^n w_{im} x_i, \text{ for } m=1,2,\dots,M$$

The best matching existing cluster is found according to maximum criteria.

$$y_j^0 = \max_{m=1,2,\dots,M} (y_m^0)$$

Step 4: The vigilance test for winning neuron j is done

$$\frac{1}{\|\mathbf{x}\|} \sum_{i=1}^n v_{ij} x_i > \rho$$

where ρ is the vigilance parameter and the norm $\|\mathbf{x}\|$ is defined for the purpose of this algorithm as follows:

$$\|\mathbf{x}\| = \sum_{i=1}^n |x_i|$$

if the vigilance test is passed, the algorithm goes to step 5. If the test is failed and top layer has more than a single active node then the algorithm goes to step 6, otherwise to step 5.

Step 5: In this step, the weight matrices are updated for neuron j passing the vigilance test. So updates are only for entries (i,j) , where $j= 1,2,\dots, M$

$$w_{ij}(t+1) = \frac{v_{ij}(t)x_i}{0.5 + \sum_{i=1}^n v_{ij}(t)x_i}$$

$$v_{ij}(t+1) = x_i v_{ij}(t)$$

After updating of weights the algorithm goes to step 2.

Step 6: The neuron is deactivated by setting $y_j=0$. This node does not participate further in the clusters search. The algorithm goes back to step 3 and attempts to find a new cluster other than j for the pattern under test.

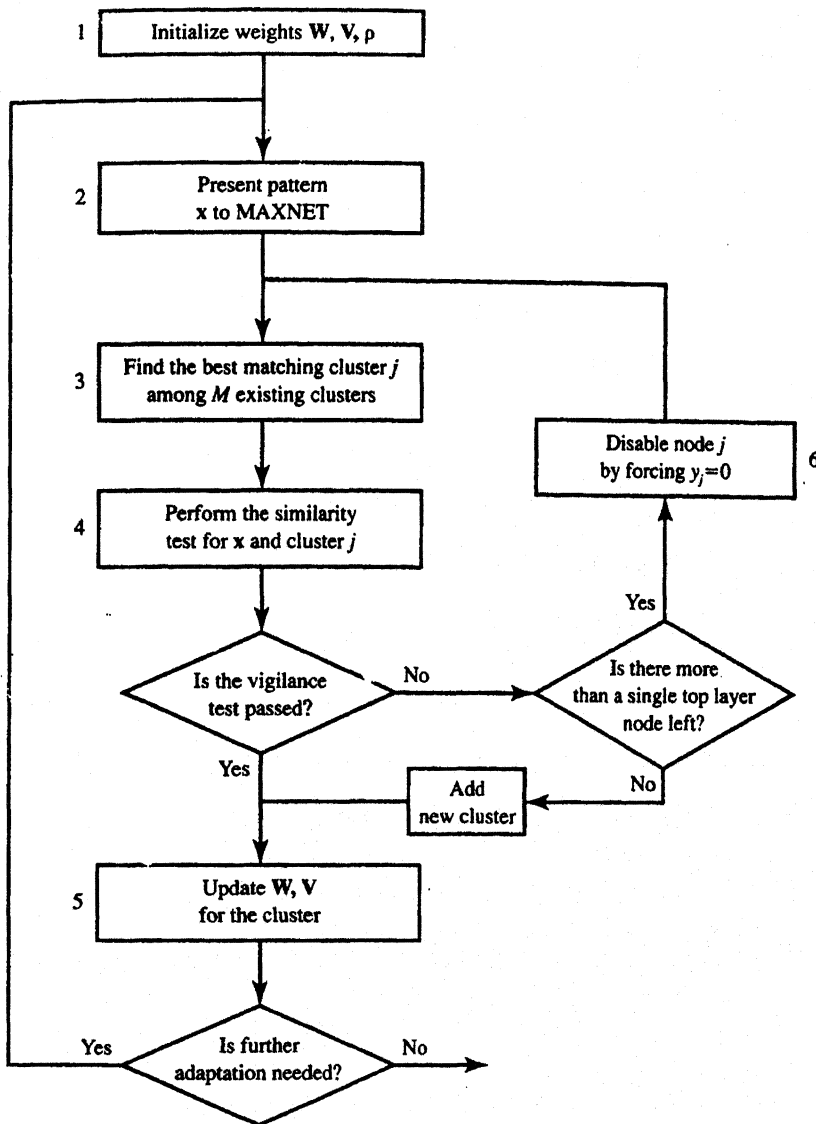


Fig 3.6: Flowchart of the ART1 Encoding Algorithm for Unipolar Binary Inputs

3.5.2 Storage Capacity

For patterns of similar shape the ART1 network is not able to classify them in two different clusters. Similar patterns are grouped into same class.

3.5.3 Result

ART1 network has been used to classify 26 alphabets. 26 neurons are taken in the top layer. Each of the 26 alphabets is stored as 49 dimensional vectors with entries 0 and 1 with vigilance threshold of 0.95 the 26 alphabets are classified in to 14 classes. Characters which are slightly similar in shape are clustered by the network in the same class. The way in which characters are classified has been shown in table 3.5. The network has been tested for different vigilance factors. It has been observed that with increase of threshold, number of classes in which these characters are classified increases. The effect has been shown in the table 3.6.

3.5.4 Merits and Demerits

Controlled discovery of clusters are one of the properties of ART1 network. The ART1 network is capable to accommodate new patterns without affecting the recall capabilities for patterns already learned. The weights for clusters already stored are not modified.

The ART1 network is not capable of classifying characters having slightly similar shape. Classification depends a lot on proper value of vigilance factor ρ .

Characters	Cluster Number
A	1
B	6
C	5
D	10
E	4
F	4
G	5

H	26
I	8
J	8
K	26
L	9
M	26
N	26
O	10
P	11
Q	14
R	6
S	12
T	8
U	10
V	13
W	14
X	15
Y	16
Z	15

**Table 3.5: Clustering of 26 alphabets by ART1 for vigilance factor
 $\rho=0.95$**

Vigilance factor	Number of Classes	Characters Classified in Same Class
.0.80	9	S; E; I; T; J; F, K; C, D, G, L, O, Q, U; N, V, W, P; X, Y, Z; A, B, H, M, R
0.85	10	A; E, F, S; C, M; I, J, T; D, G, O, Q, U; W; B, H, K, R; V; Y, Z; M, N, P, X
0.90	12	A; G; E, F; D, O, Q, U; I, J, T; C, L; P; S; V; W; N; Y, Z; B, H, K, M, R, X
0.95	14	A; E, F; C, G; B, R; I, J, T; L; D, O, U; P; S; V; Q, W; X, Z; Y; H, K, M, N
0.98	14	A; E, F; C, G; S; I, J, T; L; U; P; R; V; D, Q, W; X, Z; Y; B, H, K, M, N, O
0.99	14	A; E, F; C, G; S; I, J, T; L; U; P; R; V; D, Q, W; X, Z; Y; B, H, K, M, N, O

Table 3.6: Effect of Vigilance Factor on Classification in ART1 Algorithm

Note:”,” is used to separate the characters clustered in same class

3.6 Back Propagation Algorithm

Multilayer feed forward network can be applied to a variety of classification and recognition problems [53, 54]. In back propagation algorithm, it is not necessary to know the mathematical model of the classification of recognition problem to train and then recall information from the network. In this algorithm, if suitable network architecture is

selected and sufficient training set is presented, the network may give acceptable solution.

The back propagation algorithm allows input/output mapping within Multilayer feed forward neural networks. Input patterns are submitted during training sequentially. If classification of a submitted pattern is found to be erroneous, then the weights and threshold are adjusted so that current least mean square classification error is reduced. The input/output mapping, comparison of target and output values and adjustments are continued until all the training patterns are learned within acceptable error. During the classification phase, the trained neural network operates in a feed forward manner. The weight adjustments by the learning rule propagate backward from the output layer through hidden layer to the input layer.

3.6.1 Algorithm

Consider that there are P training pairs $\{a_1, b_1; a_2, b_2 \dots a_p, b_p\}$ a_i is the input vector and b_i is target output vector.

Step 1: Weights W and V are initialized at some random value. V is the weight matrix between input and hidden layer; W is the weight matrix between hidden and output layer.

Step 2: Pattern Z is submitted to the network and responses of neurons of hidden and output layer are calculated. y is the output hidden layer of neurons and o is the output of output layer neurons.

$$y = \Gamma[V_z]$$

$$o = \Gamma[W_y]$$

Where Γ is a nonlinear operator.

$$\Gamma[.] = \begin{bmatrix} f(.) & 0 & \dots & \dots & 0 \\ 0 & f(.) & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \dots & f(.) \end{bmatrix}$$

Step 3: The error is calculated as

$$E \leftarrow E + \frac{1}{2} \|d - o\|^2$$

Step 4: Errors δ_o and δ_y are calculated as

$$\begin{aligned} \delta_o &= \frac{1}{2} [(d_k - o_k)(1 - o_k^2)] \\ \delta_y &= w'_j \delta_o f'_y \\ f'_y &= \frac{1}{2} [1 - y_j^2] \end{aligned}$$

Step 5: weights of output layer are adjusted by

$$W \leftarrow W + \eta \delta_o y^k$$

Step 6: weights of hidden layer are adjusted by

$$V \leftarrow V + \eta \delta_y z^t$$

Step 7: If there are more patterns in the training set then the whole process from step 2 is repeated. If there are no more patterns left then it is checked whether $E < E_{\max}$. If yes then the process is complete. If no then a training cycle is started from step 2.

3.6.2 Result

The network architecture used was 35-16-26. Characters are represented by a pattern grid of 5×7 . All the 26 characters were stored as 35-dimensional vector with entries 0 and 1. The target output vector is also stored which is identity matrix of size 26×26 . An error goal of 0.01 was taken. The network

was trained for maximum 5000 epochs. After training the network was capable of recognizing all the characters correctly.

3.6.3 Merits/ Demerits

This method provides accurate input –output mapping. The advantage of this method is that each adjustment step is computed quickly. All the patterns need not be presented simultaneously. Each step is computed without finding an overall direction of the descent for the training cycle.

Multilayer feed forward neural networks trained by back propagation algorithm are slow to learn.

3.7 Quickprop

Standard Backpropagation calculates the weight change based upon the first derivative of the error with respect to the weight. If second derivative information is also available then better step of the optimum step direction can be found. Backpropagation networks are also slow to train. Quickprop is a variation of standard Backpropagation, to speed up training.

The Quickprop modification is an attempt to estimate and utilize second derivative information [Fuhrman 1988]. Quickprop requires saving the previous gradient vector as well as previous weight change. The calculation of weight change uses only information associated with the weight being updated.

$$\Delta w_{ij} = \frac{\nabla w_{ij}(n)}{[\nabla w_{ij}(n-1) - \nabla w_{ij}(n)] \Delta w_{ij}(n-1)}$$

Where $\nabla w_{ij}(n)$ = the gradient vector component associated with weight w_{ij} in step n .

$\nabla w_{ij}(n-1)$ = the gradient vector component associated with weight w_{ij} in previous step.

$\Delta w_{ij}(n-1)$ = Weight change in $(n-1)^{\text{th}}$ step.

A maximum growth factor μ is used to limit the rate of increase of the step – size like

If $\Delta w_{ij}(n) > \mu \Delta w_{ij}(n-1)$,

Then $\Delta w_{ij}(n) > \mu \Delta w_{ij}(n-1)$

Fahlman suggested an empirical value 1.75 for μ .

There are some complications in this method. First is the step-size calculation that requires a previous value, which is not available at the time starting. This is overcome by using standard back propagation method for the weight adjustment. The gradient descent weight change is given by,

$$w_{ij}(n+1) = w_{ij}(n) - \eta \nabla w_{ij}$$

Value of η is taken suitably small.

Second problem is that the weight values are unbounded. They become so large that they cause overflow in the computer. Its solution is that each calculated slope is multiplied by a factor less than 1.0, which reduces the rate of increase of the weight.

3.7.1 Result

Quickprop algorithm has been tested for 26 characters. The network has been trained for a 80 epochs. The most suitable network architecture found was 49-16-26. All the 26 characters in the form of 49-dimensional vectors and target output as 26×26 identity matrix are given during training. The network gives output equal to the target output. The value of learning rate used is 0.8 and a momentum factor of 0.1 is taken. The no. of characters

recognized correctly varies with a learning rate. The effect is shown in table 3.7. There is no effect of varying momentum factor. Varying momentum factor between 0.1-0.9 for learning factors 0.1-1.0 has tested this.

Learning Rate	Number of Characters Recognized correctly
0.1	1
0.2	14
0.3	24
0.4	19
0.5	17
0.6	25
0.7	6
0.8	26
0.9	14
1.0	13

**Table 3.7: Effect Of Learning Rate On Number Of Characters
Recognized Correctly.**

3.7.2 Merits and Demerits

Quickprop reduces the training time of back propagation it hardly takes 2 or 3 seconds.

The selection of network architecture should be carefully done. Its performance is highly problem dependent. It requires several trials to set acceptable values of parameters. One problem is that weight values are unbounded so that they cause overflow problem in the computer.

3.8 Kohonen Self-Organizing Map

Kohonen network is based on unsupervised learning of clusters. The term self-organization means the ability to learn and organize information by it self. Thus a self-organization network performs unsupervised learning.

Kohonen network consists of single layer of neurons plus an input layer. Each neuron receives input from other neurons within layers. In Kohonen network, the weights are initialized to some random values. The learning in this network is winner take all learning (WTA). The network to be trained is called the kohonen network [55]. The neuron with largest activation value is declared as winner. Only this neuron gives an output, all other neurons are suppressed to zero activation level. The output of neuron is given as inhibitory input to other neurons but as excitatory to its neighbors. Thus weight is updated not only of winner neuron but also of other's. This type of competition within a layer is called lateral inhibition. The neighborhood size decreases with learning. The number of learning neuron also decreases in each iteration and finally only the winner neuron learns.

3.8.1 Algorithm

Step 1: The training set consists of p patterns are presented to the input.

$$\{a_1, a_2, \dots, a_p\}$$

Each pattern is n-dimensional.

Step 2: Weight Matrix is initialized as

$$W = \begin{bmatrix} w_1^t \\ w_2^t \\ \vdots \\ w_p^t \end{bmatrix}, \text{ Where } w_i = \begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{in} \end{bmatrix} \text{ for } i=1,2,\dots,p$$

Step 3: Neighborhood distance of neurons is fixed and neighborhood matrix is taken.

Step 4: The network is trained using weight matrix W , neighborhood matrix M and the stored patterns.

Step 5: For each of the training pattern, winner neuron is found. Training pattern is presented to the network one by one. The weight adjustment is done by selecting w_i such that

$$\|a - \hat{w}_m\| = \min\{\|a - \hat{w}_i\|\} \quad \text{for } i=1,2,\dots,p$$

The index m denotes the winning neuron number corresponding to \hat{w}_m , which is closest to a .

Step 6: The weight of the winner neuron is updated such that

$$\Delta w'_m = \alpha(a - w_m)$$

Where α is learning constant.

The remaining weight vector is left as it is. So

$$w_m^{k+1} = w_m^k + \alpha^k(a - w_m^k)$$

$$w_i^{k+1} = w_i^k \quad \text{for } i \neq m$$

α is reduced during training step and the learning slows down.

Step 7: The test vector x is presented to the network after training the network is in winner-take-all modes. The response is computed from

$$y = \Gamma[W_x]$$

Γ is an operator with diagonal entries which operates on components of Wx . Dimension of y is p .

Step 8: Largest output found as

$$y_m = \max(y_1, y_2, \dots, y_p)$$

y_m is set to 1 and for $i \neq m$ $y_i = 0$. So, the test vector is identified as belonging to cluster m .

3.8.2 Result

Training set consists of 26 characters each represented in the form of pattern grid of 7×7 . 1 is used to represent an on pixel and zero for an off pixel. These 26 characters are classified into 23 classes using 42 neurons. The network has been trained for 5000 epochs. It has been observed that no. of neurons influences the no of classes. The effect of increasing numbers of neurons is given in table 3.8.

Number of Neurons	Number of Classes
26	20
28	20
30	21
32	22
34	22
36	23
38	23
40	23
42	24
44	24

Table 3.8 Effect of number neuron on the classification in Kohonen network

3.8.3 Merits

The neuron's activation function does not play any influence on the performance of the network.

3.9 Neocognitron

The Neocognitron is self organized by unsupervised learning and acquires the ability for correct pattern recognition. For self-organization of network, the patterns are presented repeatedly to the network [56, 57]. It is not needed to give prior information about the categories to which these patterns should be classified. The Neocognitron itself acquires the ability to classify and correctly recognize these patterns, according to the differences in their shapes.

The network has a Multilayer structure. Each layer has two planes. One called s-plane, consists of s units. Other the c-plane consists of c units. The units can have both excitatory and inhibitory connections. The network has forward connections from input layers to output layer and backward connections from the output layer to the input layer. The forward signals are for pattern classification and recognition. The backward signals are for selective attention, pattern segmentation and associated recall.

3.9.1 Algorithm

Let $u(1), u(2), \dots, u(N)$ be the excitatory inputs and v be the inhibitory input. The output of the s-cell is given by [58]

$$w = \varphi \left[\frac{1 + \sum_{v=1}^N a(v)u(v)}{1 + bv} - 1 \right]$$

Where $a(v)$ and b represent the excitatory and inhibitory interconnecting coefficients respectively.

The function $\varphi[]$ is defined by

$$\begin{aligned} \varphi[x] &= x & x \geq 0 \\ 0 & & x < 0 \end{aligned}$$

Let e be the sum of all excitatory inputs weighted with interconnecting coefficients and h is the inhibitory input multiplied by the interconnecting coefficients.

$$e = \sum_{v=1}^N a(v)u(v)$$

$$h = b.v$$

Therefore w can be written as,

$$w = \phi \left[\frac{1+e}{1+h} - 1 \right] = \phi \left[\frac{e-h}{1+h} \right]$$

When $h \ll 1$, $w = \phi(e-h)$

In Neocognitron, the interconnecting coefficients $a(v)$ and b increases as learning progresses. If the interconnecting coefficient increases and if $e \gg 1$ and $h \gg 1$,

$$w = \phi \left[\frac{e}{h} - 1 \right]$$

Where the output depends on the ratio e/h , not on e and h . Therefore, if both the excitatory and inhibitory coefficients increase at the same rate then the output of the cell reaches a certain value.

Similarly the input and output characteristics of c-cell is given by,

$$\psi[x] = \begin{cases} \frac{x}{\alpha + x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Where α is a positive constant which determines the degree saturation of the output.

The output of the excitatory cells s and c are connected only to the excitatory input terminals of the other cells. V_s and V_c cells are inhibitory cells, whose output is connected only to the inhibitory input terminals of

other cells. A V_s -cell gives an output, which is proportional to the weighted arithmetic mean of all its excitatory inputs. A V_c -cell also has only inhibitory inputs but its output is proportional to the weighted root mean square of its inputs. Let $u(1), u(2), \dots, u(N)$ be the excitatory inputs and $c(1), c(2), \dots, c(n)$ be the interconnecting coefficients of its input terminals. The output w of this V_c -cell is defined by

$$w = \sqrt{\sum_{v=1}^N c(v)u^2(v)}$$

3.9.2 Storage Capacity

By increasing number of planes in each layer the capacity of the network can be increased. But by increasing no of planes in each layer, the computer cannot simulate because of the lack of memory capacity.

3.9.3 Results

A nine layered network $U_0, U_{s1}, U_{c1}, U_{s2}, U_{c2}, U_{s3}, U_{c3}, U_{s4},$ and U_{c4} is taken. So the network has 4 stages proceeded by an input layer. There are 21 cell planes in each layer $U_{s1}-U_{s4}$, and 20 cell planes in $U_{c1}-U_{c4}$. The parameter are listed in table 3.9. As can be seen from the table total number of c-cells in layer U_{c4} is 20 because each c-plane has only one cell. The number of cells in a connectable area is always 5×5 for every s-layer. Hence number of excitatory input to each s-cell is 5×5 in layer U_{s1} , and $5 \times 5 \times 20$ in layers U_{s2}, U_{s3} and U_{s4} because these layers are preceded by c-layers consisting of 20 cell planes in each.

During learning five training patterns 0,1,2,3 and 4 were presented repeatedly to the input layer. After repeated presentation of these five patterns the Neocognitron acquired the ability to classify these patterns according to the difference in their shape. Each pattern is presented five times. It has been observed that a different cell responds to each pattern.

Layer	Number of Excitatory Cells	Number of Excitatory input Interconne ctions per cell	Size of an s-column (No of cells per s- column)	r_l	q_l
U_0	19×19				
U_{s1}	19×19×21	5×5	5×5×20	4.0	1.0
U_{c1}	15×15×20	5×5			
U_{s2}	13×13×21	5×5×20	5×5×20	1.5	12.0
U_{c2}	11×11×20	5×5			
U_{s3}	9×9×21	5×5×20	5×5×20	1.5	12.0
U_{c3}	7×7×20	5×5			
U_{s4}	3×3×21	5×5×20	3×3×20	1.5	12.0
U_{c4}	20	3 × 3			

Value of $\alpha = 0.75$

Table 3.9: Architecture and values of parameters used in Neocognitron

3.9.4 Demerits

The Network architecture is very complex. It involves a large no of neurons.

3.9.5 Conclusions

The algorithms of various neural networks that can be used for character recognition has been studied. The capacity of storing and correctly recognizing all the 26 English alphabets has also been found. It has been observed that only three algorithms are capable of storing all the 26 alphabets. These three are: (1) Hamming Network and MAXNET, (2)

Backpropagation Algorithm and (3) Quickprop. Among these the performance of Backpropagation and Quickprop largely depends upon network architecture. Hamming network does not have this problem. However, it only gives the class index, not the entire prototype vector.

Chapter 4: Different Analysis Criterion

Chapter 4: Different Analysis Criterion

4.1 Introduction

The various neural network algorithms differ in their learning mechanism. Some networks learn by supervised learning and in some learning are unsupervised. The capacity of storing patterns and correctly recognizing them differs for different networks. Although if some of the networks are capable of storing same number of patterns, they differ in complexity of their architecture. Total number of neurons needed to classify or recognize the patterns are different for various algorithms. Total number of unknowns is also different for various algorithms. The networks are compared on the basis of these. In this chapter, the performance of various algorithms under different criteria is presented.

These criteria are:

1. **Noise in Weight:** Noise has been introduced in two ways,
 - (i) By adding random numbers to the weight
 - (ii) By adding a constant number to all weights of the network
2. **Noise in Input:** The network is trained with characters without noise. Then by presenting each of the character, the network is tested. It has been observed that the number of characters recognized correctly differs for various algorithms. Noise is introduced randomly to the input vector at the time of testing and its effect has been observed on various algorithms. This has been done by adding random numbers to the test vector.
3. **Loss of Connection:** In every network neurons are interconnected by some weight. If we loose any connection by equating the weight to zero,

then how it is going to influence the network's classification or recognition ability has been studied. For every network, the number of connections that can be lost such that the classification or recognition is not affected has been found. For some network algorithm, it has been observed that the no varies from character to character

4. Missing Information from Test Pattern: The characters are presented in the form of pattern grid. If some of information is missed than how it is going to influence has been observed. The missing information means that some of the on pixels in the pattern grid are made off. For all the algorithms, the maximum number of pixels that can be made of so that the character is recognized correctly is found. This number varies from character to character for each algorithm. Which pixel is made off i.e. from where information is missing also matters?

5. Adding Information to Test Pattern: Adding information means some of the off pixels in the pattern grid are made on. The number of pixels that can be made on so that the character is recognized correctly is found. Like missing information, this also varies from character to character for each algorithm.

4.2 Effect of Noise in Weight on Various Algorithms

4.2.1 When Weights Are Varied Randomly

(i) Bidirectional Associative Memory

When random numbers are added in the weight matrix, only character A was recognized correctly.

When random numbers are added after dividing by 10 then all the 6 characters are recognized correctly.

(ii) Hopfield Auto associative Memory

When random numbers are added in the weight matrix, the characters are not recognized at all. Only dots appear for every character.

When 100 divide random numbers then all the 6 characters are recognized correctly.

(iii) Feature Recognition Neural Network

When random numbers are added in the weight matrix, the network does not work.

(iv) Hamming Network and MAXNET

When noise is introduced by adding random numbers to the weights connecting inputs to neurons of Hamming network, all the characters are recognized correctly. But if random numbers are added to the weights connecting neurons of Hamming network and MAXNET, MAXNET does not work. Hamming network gives correct result but the final out put of MAXNET is wrong.

(v) Backpropagation Algorithm

When random numbers are added to the weight matrices of all layers, 21 characters are recognized correctly. If at one time, noise is introduced in weight only one layer then all the characters is recognized correctly.

(vi) Quickprop

When random numbers are added in the weight between input neurons and hidden neurons at 80 epochs not a single character is recognized correctly. When number of epochs is increased to 85, 11 characters are recognized correctly. For 100 epochs 17 characters are recognized correctly.

When random numbers are added to the weights connecting neurons of hidden layer and output layer for 80 epochs, actual output is equal to the

target output for 23 characters. When epochs are increased to 85, 24 characters are recognizing correctly.

When random numbers are added to both the layers at the same time, variation in number of characters recognized correctly with increase of number of epoch has been shown in table 4.1 below,

Number of Epochs	Characters Recognized Correctly
80	8
85	12
90	13
95	16

Table 4.1: Effect of Number of Epochs on the Number of Characters Recognized Correctly

Algorithms	Range of % increase of weights for which does work properly	Range of % increase of weight for which there is no effect
BAM	0.58-49.7	0.058-4.97
Hopfield Auto associative Memory	0.015-99.93	0.0015-0.99
Hamming Network and MAXNET	N.A.	0.015-99.93
Backpropagation Algorithm	0.0485-53.85	0-10.77
Quickprop	0.12-55.46	N.A.
ART1	0-7.5	N.A.
Kohonen SOM	N.A.	0-70

Table 4.2: For each algorithm: effect of % increase in weight

(vii) ART1

When random numbers are added in weights 26 characters are classified into 7 classes.

(viii) Kohonen SOM

There is no effect of adding random numbers to the weights. With 42 neurons 26 characters are clustered into 23 classes.

(xi) Neocognitron

It does not work properly after introducing noise in weights. Table 4.2 shows the range of percentage increase in weight for each algorithm.

4.2.2 By adding constant number to all the weight of network

(i) Bidirectional Associative Memory

When 0.1 is added to all the weights characters A, L, X and P are recognized correctly. I and M are never recognized even if a very small no. (0.0001) is added.

(ii) Hopfield Auto associative Memory

When 0.0065 is added to all the weights, characters stored in the network are recognized correctly.

(iii) Feature Recognition Neural Network

It does work even after adding very small number also.

(iv) Hamming Network and MAXNET

All the characters are recognized correctly, if we add 3.5 to the weights between input layer and neurons of hamming net part. But if we add any number to the weights connecting neurons of hamming net and MAXNET then the network does not work at all.

(v) Backpropagation Algorithm

There are three weight matrices. Weight connecting inputs to the input layer, input layer to hidden layer, and hidden layer to output layer.

If we add 0.08 to all the weights, then it does not have any effect on the network. If we increase weights of only one layer at one time then a large number can be added. It has been observed that if 0.4 is added to any of three weights at one time, then the network works as before.

(vi) Quickprop

If we add 0.0002 to all the weights then for all the 26 characters we get output equal to the target output. If this number is increased then for some of the characters we do not get output same as target output.

(vii) ART1

When adding 0.01 increase weight, then we get 14 numbers of classes as before. Only effect is that for characters H, K, M and N neuron number 25 wins instead of 26.

(viii) Kohonen SOM

If we add 0.5 to all the weights, then it does not have any effect, we get 23 classes as before.

4.2.3 Effect of Noise in Inputs on Various Algorithms

Noise is introduced in the input by adding random numbers.

(i) Bidirectional Associative Memory

When random numbers is added to the characters A, L, I, X, M and P the network is able to correctly recognize only A, L, X and P.

(ii) Hopfield Auto associative Memory

Hopfield network recognizes correctly all the stored characters even after introducing noise at the time of testing.

(iii) Feature Recognition Neural Network

This network does not work after introducing noise randomly in the test vector.

(iv) Hamming Network and MAXNET

After introducing noise in test vector, the network correctly recognizes all the 26 characters.

(v) ART1

When noise is introduced in the input vector at time of testing, all the characters are clustered in to same class.

(vi) Kohonen SOM

26 characters are classified into 18 classes, when noise is introduced by adding random number to the vectors representing them.

4.2.4 Loss of Connection

In the network, neurons are interconnected and every interconnection has some interconnecting coefficient called weight. If some of these weights are equated to zero then how it is going to effect the classification or recognition, is studied under this section. The number of connections that can be removed such that the network performance is not affected has also been found out for each algorithm.

- (i) Bidirectional Associative Memory:** If the weight connecting 4 input neurons to all the output neurons is equated to zero, network performance is not affected.
- (ii) Hopfield Auto associative Memory:** If connection of input neuron's to all the output neuron is removed, and the pixel corresponding to that neuron number is off than it makes no difference. But if that pixel is on, in the output that becomes off.

- (iii) **Feature Recognition Neural Network:** After loosing 234 connections between layer one and layer two the network performs as before.
- (iv) **Hamming Network and MAXNET:** If we loose connection between Hamming net to MAXNET, than the network does not work. But after loosing a few connections connecting input to hamming network, then the characters are classified correctly. For some character like Q, it is recognized correctly even after loosing connection of 46 input to all the neurons of Hamming network. The number of redundant connection for each of the character is shown in the table 4.3.

Character	Connection We Can Loose
C	26
T	34
L, M	35
J,S,W	37
F,O	38
A,B,E,K,N,V	39
D,G,I,R,U,Z	40
H,P,X,Y	41
Q	46

Table 4.3: Number of Redundant Connection for Each Character

- (v) **Backpropagation Algorithm:** There are three sets of weights. First connecting input to neurons of input layer. Second connecting input layer neuron to the hidden layer neurons. Third connecting hidden layer neurons to output layer neurons. If the weights connecting 15 input to 35 neurons of input layer, 15 neurons of input layer to 15 neurons of

hidden layer and 8 neurons of hidden layer to all the 26 neurons of output layer are removed, then also the network recognizes all the characters correctly. So total numbers of redundant connections are 958.

- (vi) **Quickprop:** There are two sets of weights. First connecting input neurons to hidden neurons to output neurons. If the weights connecting 49 input neurons to 15 hidden neurons and weights connecting 15 hidden neurons to 2 output neurons is equated to zero, then for 25 characters actual output is equal to the target output.
- (vii) **ART1:** If all the weights of the forward network are equated to zero, then it does not make any difference. All the 26 characters are clustered as before.
- (viii) **Kohonen SOM:** If all the weights connecting input to neurons are made zero, then it classified 26 characters in to 23 classes. So there is no effect of making weights zero.
- (ix) **4.2.5 Missing Information:** Missing information means some of the on pixels in pattern grid are made off. For each of the algorithm how many information we can miss so that the characters can be recognized correctly varies from character to character. We cannot switch off pixel from any place. Which pixel is being switched also matters. In each algorithm, the way in which information is missed is shown in the appendix A. For few characters table 4.4 shows the number of pixels that can be switched off for all the stored characters in various algorithms. The comparison is done for Bidirectional associative memory, Hopfield auto associative memory, feature recognition neural network, hamming network and MAXNET, ART1, and Kohonen network. From table 4.4 it is clear that ART1 correctly classifies after

loosing maximum number of information. For some characters it puts them in the same cluster as before missing information even after making 12 to 13 on pixels off. Hamming Net and MAXNET can also classify correctly after loosing a large number of information.

4.2.6 Adding Information: Adding information means some of the off pixels in the pattern grid are made on. In this section, the classification or recognition ability of networks after adding information is studied. For every algorithm the number of pixels that can be made on varies from character to character. Which pixels are made on also matters? Appendix shows the pattern grid of the test vector given as input to different algorithms after adding information. Table no. 4.5 shows detailed description about the number of pixels that can be made on for all the characters that can be stored in various networks. The comparison is done for Bidirectional associative memory, Hopfield auto associative memory, feature recognition neural network, hamming network and MAXNET, ART1, and Kohonen network. From the table 4.5, it is clear that FRNN correctly classifies after adding maximum number of information. For some characters it correctly classifies even if 31-32 extra pixels are made on.

Algorithms Character	BAM	Hopfield Auto Associative Memory	FRNN	Hamming Network and MAXNET	ART1	Kohonen SOM
A	6		1	6	13	7
B			2	12	5	5
C			4	7	8	5
D			4	5	2	5
E			0	10	13	9
F			1	11	12	6
G			3	9	11	4
H			2	12	10	8
I	5	5	5	5	9	6
J			7	9	7	8
K			4	8	8	5
L	4	6	4	5	11	6
M	4	5	2	5	12	6
N			2	11	1	3
O		7	2	7	0	6
P	2		3	12	12	10
Q			6	12	12	7
R			3	12	12	9
S			2	12	12	4
T		7	6	6	10	6

U			3	11	11	6
V			3	10	10	8
W			3	12	12	9
X	5		8	7	9	5
Y		2	6	2	6	4
Z			2	8	10	7

Table 4.4: Missing Information: No Of Pixels That Can Be Made Off In Different Algorithms

Algorithms Character	BAM	Hopfield Auto Associative Memory	FRNN	Hamming Network and MAXNET	ART1	Kohonen SOM
A	6		10	6	2	8
B			18	12	1	11
C			18	14	2	8
D			23	14	1	10
E			19	14	2	14
F			20	14	3	10
G			19	13	4	8
H			24	16	17	14
I	11	11	14	12	10	7
J			21	19	3	14
K			22	15	15	5
L	5	5	7	5	2	12

M	11	14	19	14	7	8
N			24	15	16	10
O			18	5	5	9
P	10		29	18	4	18
Q			20	15	4	15
R			20	14	1	14
S			27	16	2	13
T		9	14	9	4	10
U			31	19	1	17
V			31	19	2	18
W			30	14	9	14
X	9		11	8	4	8
Y		10	14	10	4	8
Z			32	19	2	11

**Table 4.5: Adding Information: No Of Pixels That Can Be Made On In
Different Algorithms**

Chapter 5: Conclusions

Chapter 5 Conclusions

5.1 General Conclusions

A detailed analysis of 9 algorithms BAM, Hopfield, FRNN, Hamming network and MAXNET, Backpropagation, Quickprop, ART1, Kohonen and Neocognitron has been performed for the pattern recognition (English Alphabets A-Z). The storage capacity of each algorithm is different. The near optimal network architecture of each of these algorithms was found, so that it can correctly classify or recognize the stored pattern.

The Algorithms were compared on the basis of following criteria:

1. Total number of neurons needed
2. Total number of unknowns to be computed
3. Storage capacity
4. Noise in weight
 - (i) By adding random numbers to the weight
 - (ii) By adding a constant number to all the weights
5. Noise in input
6. Missing information from the test vector
7. Adding information to the test vector
8. Number of redundant connections

BAM and Hopfield auto associative memory have capacity limitations. Both the network can store and correctly recognize only six characters, with the condition that the characters should not be slightly similar in shape. FRNN can recognize 23 characters correctly. It recognizes O and Q as C. It can not differentiate R from P. It performs well. But the number of neurons and the number of connection are much greater when compared to other algorithms. Its architecture is much simpler in comparison to Neocognitron for doing similar task. Neocognitron can be design to correctly recognize all the 26 characters. But it involves a large number of neurons and connections. This network is very complex. A latest Neocognitron is developed which recognizes 35 patterns. It involves 70045 neurons. So the complexity can be imagined. Only 3 networks are found to correctly recognize or classify all the 26 characters. These 3 are ,

- (i) Hamming network and MAXNET
- (ii) Backpropagation Algorithm
- (iii) Quickprop

Among them in slandered Backpropagation and Quickprop algorithms, we have two suitable select the network architecture to perform the above task. The number of hidden neurons plays a great role in the performance of these two networks. Hamming network does not have this problem. However, the Hamming network retrieves only the class index and not the entire prototype vector. This network is not able to restore any of the pattern vector entries. When compared on the basis of number of neurons, it has been observed that Hamming network involves minimum number of neurons. It requires only 52 neurons in comparison to Backpropagation, which needs 76 neurons and Quickprop that involves 91 neurons. FRNN requires 2371 neurons. But the total number of unknowns is least for quick prop. It requires 1200 unknowns

to be determined in comparison to Hamming network, which requires 1976, and Backpropagation, which requires 2216 unknowns. When noise is introduced in the test vector by adding random numbers, Hopfield auto associative memory and Hamming network recognizes all the stored characters correctly. So we can say that there is no effect on recognition ability of these networks by noisy inputs. BAM can correctly recognize 4 out of stored 6 characters. ART1 puts all the characters in the same class after introducing noise. FRNN does not work after adding random numbers to the test vector.

ART1 performs best after missing information from the test vector. Hamming network also performs well after missing information. FRNN performs best after adding information in the test vector. It correctly classifies even after making on 31-32 off pixels. Kohonen network also performs well after adding a large number of information.

The performance of 9 algorithms has been studied under six criteria. It has been observed that a certain algorithm performs best under a particular criterion. The algorithms have also been compared based on the number of neurons and the number of unknowns to be computed. The detailed description is shown in table 4.6.

The performance of 9 algorithms has been studied under six criteria. It has been observed that a certain algorithm performs best under a particular criterion. The algorithms have also been compared based on the number of neurons and the number of unknowns to be computed. The detailed description is shown in table 4.6.

Algorithms	Algo 1	Algo 2	Algo 3	Algo 4	Algo 5	Algo 6	Algo 7	Algo 8	Algo 9
Criteria									
Number of Neurons	55	98	2366	52	76	91	75	42	21301
Number of Unknowns	490	2401	21294	1976	2216	1200	2548	2058	V.L.
Capacity	6	6	23	26	26	26	26 Ch. In 14 classes	26 Ch. In 23 classes	5
Effect of Noise in Weight (Random No. Added)	Not recognizes	Not recognizes	Not works	No effect	21	8 is recognized	26 Ch. In 7 classes	No effect	No effect
Effect of Increase of Weight	0.1	0.0065	Not works	3.5	0.08	0.0002	0.01	0.05	N.S.
Noise in Input	4 is recognized	No effect	Not works	No effect	N.S.	N.S.	All in same class	26 Ch. In 18 classes	N.S.
Range of No. of Pixels that can made off	2-6	2-7	0-8	2-12	N.A.	N.A.	0-13	3-10	N.S.
Range of	5-11	5-14	7-32	5-19	N.A.	N.A.	1-17	5-18	N.S.

No. of Pixels that can made on									
No of Connection we can loose (wt=0)	15	49	234	676	958	765	No effect it all wts=0	No effect it all wts=0	N.S.

**Table 4.6: Performance of Various Algorithms under Different
Criterion**

In table 4.6 the used notation stands for:

Algo1: Bidirectional Associative Memory

Algo2: Hopfield Auto Associative Memory

Algo3: Feature Recognition Neural Network

Algo4: Hamming Network and MAXNET

Algo5: Backpropagation algorithm

Algo6: Quick Prop

Algo7: Adaptive Resonance Theory 1

Algo8: Kohonen Self Organizing Map

Algo9: Neocognitron

N.A.: Not Applicable

N.S.: Not studied

5.2 Recommendations for Future Work

- One can look at the capability of enhancing the capacity of the networks for the pattern recognition.
- The performance of various algorithms can be studied by varying style of presentation of patterns. Varying fonts can change the style.
- One can also look at the performance of networks for handwritten patterns.

Appendix A

A1. Character Presentation

A

o	o	*	*	*	o	o
o	*	o	o	o	*	o
*	o	o	o	o	o	*
*	*	*	*	*	*	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*

B

*	*	*	*	*	*	o
*	*	o	o	o	*	*
*	o	o	o	o	o	*
*	*	*	*	*	*	o
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	*	*	*	*	*	o

C

o	*	*	*	*	*	o
*	o	o	o	o	o	*
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	*
o	*	*	*	*	*	o

D

*	*	*	*	*	*	o
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	*	*	*	*	*	o

E

*	*	*	*	*	*	*
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	*	*	*	*	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	*	*	*	*	*	*

F

*	*	*	*	*	*	*
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	*	*	*	*	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o

G

o	*	*	*	*	*	o
*	o	o	o	o	o	*
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	*	*
*	o	o	o	o	o	*
o	*	*	*	*	*	o

H

*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	*	*	*	*	*	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*

I

o	*	*	*	*	*	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	*	*	*	*	*	o

J

o	*	*	*	*	*	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
*	o	o	*	o	o	o
o	*	o	*	o	o	o
o	o	*	*	o	o	o
o	o	o	*	o	o	o

K

*	o	o	o	o	*	o
*	o	o	o	*	o	o
*	o	o	*	o	o	o
*	*	*	o	o	o	o
*	o	o	*	o	o	o
*	o	o	o	*	o	o
*	o	o	o	o	*	o

L

*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	*	*	*	*	*	*

M

*	o	o	o	o	o	*
*	*	o	o	o	*	*
*	o	*	o	*	o	*
*	o	o	*	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*

N

*	o	o	o	o	o	*
*	*	o	o	o	o	*
*	o	*	o	o	o	*
*	o	o	*	o	o	*
*	o	o	o	*	o	*
*	o	o	o	o	*	*
*	o	o	o	o	o	*

O

○	*	*	*	*	*	○
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*
○	*	*	*	*	*	○

P

*	*	*	*	*	*	○
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	*	*	*	*	*	○
*	○	○	○	○	○	○
*	○	○	○	○	○	○

Q

○	*	*	*	*	*	○
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	*	○	*
*	○	○	○	○	*	*
○	*	*	*	*	○	*

R

*	*	*	*	*	*	○
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	*	*	*	*	*	○
*	○	○	*	○	○	○
*	○	○	○	*	○	○
*	○	○	○	○	*	○

S

o	*	*	*	*	*	o
*	*	o	o	o	o	o
*	o	o	o	o	o	o
*	*	*	*	*	*	o
o	o	o	o	o	o	*
o	o	o	o	o	o	*
o	*	*	*	*	*	o

T

*	*	*	*	*	*	*
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o

U

*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
o	*	o	o	o	*	o
o	o	*	*	*	o	o

V

*	o	o	o	o	o	*
*	o	o	o	o	o	*
o	*	o	o	o	*	o
o	*	o	o	o	*	o
o	o	*	o	*	o	o
o	o	*	o	*	o	o
o	o	o	*	o	o	o

W

*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	*	o	o	*
*	o	*	o	*	o	*
*	*	o	o	o	*	*
*	o	o	o	o	o	*

X

*	o	o	o	o	o	*
o	*	o	o	o	*	o
o	o	*	o	*	o	o
o	o	o	*	o	o	o
o	o	*	o	*	o	o
o	*	o	o	o	*	o
*	o	o	o	o	o	*

Y

*	o	o	o	o	o	*
o	*	o	o	o	*	o
o	o	*	o	*	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o

Z

*	*	*	*	*	*	*
o	o	o	o	o	*	o
o	o	o	o	*	o	o
o	o	o	*	o	o	o
o	o	*	o	o	o	o
o	*	o	o	o	o	o
*	*	*	*	*	*	*

A2: For Each Algorithm: Character Presentation after Missing Information

A.2.1 Bidirectional Associative Memory

A

o	o	o	*	o	o	o
o	o	o	o	o	*	o
*	o	o	o	o	o	o
*	*	o	o	*	*	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*
*	o	o	o	o	o	*

I

o	*	o	*	o	*	o
o	o	o	*	o	o	o
o	o	o	o	o	o	o
o	o	o	*	o	o	o
o	o	o	o	o	o	o
o	o	o	*	o	o	o
o	*	*	o	*	*	o

L

○	○	○	○	○	○	○
*	○	○	○	○	○	○
○	○	○	○	○	○	○
*	○	○	○	○	○	○
*	○	○	○	○	○	○
*	○	○	○	○	○	○
○	○	*	*	*	*	*

M

*	○	○	○	○	○	○
*	○	○	○	○	*	*
*	○	*	○	○	○	*
*	○	○	*	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	○
*	○	○	○	○	○	*
○	○	○	○	○	○	○
○	○	*	○	*	○	○
○	○	○	○	○	○	○
○	○	*	○	*	○	○
○	○	○	○	○	○	○
*	○	○	○	○	○	*

X

A.2.2 Hopfield Auto associative Memory

I

○	*	○	*	○	*	○
○	○	○	*	○	○	○
○	○	○	○	○	○	○
○	○	○	*	○	○	○
○	○	○	○	○	○	○
○	○	○	*	○	○	○
○	*	○	*	○	*	○

L

○	○	○	○	○	○	○
*	○	○	○	○	○	○
○	○	○	○	○	○	○
*	○	○	○	○	○	○
*	○	○	○	○	○	○
*	○	○	○	○	○	○
*	*	○	○	○	○	*

M

○	○	○	○	○	○	○
*	○	○	○	○	○	*
*	○	*	○	*	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*

O

○	○	○	*	○	○	○
*	○	○	○	○	○	*
*	○	○	○	○	○	*
○	○	○	○	○	○	*
*	○	○	○	○	○	*
*	○	○	○	○	○	*
○	○	*	*	*	○	○

T

*	o	o	o	o	o	*
o	o	o	o	o	o	o
o	o	o	*	o	o	o
o	o	o	o	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o

Y

*	o	o	o	o	o	*
o	*	o	o	o	*	o
o	o	*	o	*	o	o
o	o	o	o	o	o	o
o	o	o	*	o	o	o
o	o	o	o	o	o	o
o	o	o	*	o	o	o

A.2.3 Feature Recognition Neural Network

I

o	o	*	*	o	*	*	o	o
o	o	o	o	o	o	o	o	*
o	o	o	o	*	o	o	o	o
o	o	o	o	o	o	o	o	o
o	o	o	o	*	o	o	o	o
o	o	o	o	o	o	o	o	o
o	o	o	o	*	o	o	o	o
o	o	o	o	*	o	o	o	o
o	o	*	*	o	*	*	o	o

J

```

o * * o o o * * o
o o o o o o o o o
o o o o * o o o o
o o o o * o o o o
o o o o o o o o o
* o o o * o o o o
o o o o * o o o o
o o * o o o o o o
o o o o * o o o o

```

L

```

o o o o o o o o o
* o o o o o o o o
o o o o o o o o o
* o o o o o o o o
* o o o o o o o o
* o o o o o o o o
* o o o o o o o o
* o o o o o o o o
o o * * * * * * *

```

M

```

* o o o o o o o *
* o o o o o o * *
* o * o o o * o *
* o o o o * o o *
* o o o * o o o *
* o o o o o o o *
* o o o o o o o *
* o o o o o o o *
* o o o o o o o *

```

X

*	o	o	o	o	o	o	o	*
o	o	o	o	o	o	o	o	o
o	o	*	o	o	o	*	o	o
o	o	o	o	o	o	o	o	o
o	o	o	o	o	o	o	o	o
o	o	o	o	o	o	o	o	o
o	o	*	o	o	o	o	o	o
o	*	o	o	o	o	o	*	o
*	o	o	o	o	o	o	o	*

Y

*	o	o	o	o	o	o	o	*
o	o	o	o	o	o	o	o	o
o	o	*	o	o	o	*	o	o
o	o	o	o	o	o	o	o	o
o	o	o	o	*	o	o	o	o
o	o	o	o	o	o	o	o	o
o	o	o	o	o	o	o	o	o
o	o	o	o	*	o	o	o	o
o	o	o	o	*	o	o	o	o

A.2.4 Hamming Network and MAXNET

I

o	*	*	o	*	*	o
o	o	o	o	o	o	o
o	o	o	*	o	o	o
o	o	o	o	o	o	o
o	o	o	*	o	o	o
o	o	o	o	o	o	o
o	*	*	o	*	*	o

L

○	○	○	○	○	○	○
*	○	○	○	○	○	○
○	○	○	○	○	○	○
*	○	○	○	○	○	○
*	○	○	○	○	○	○
*	○	○	○	○	○	○
○	○	○	*	*	*	*

P

*	○	○	○	○	*	○
○	○	○	○	○	○	*
*	○	○	○	○	○	○
○	○	○	○	○	○	○
○	*	*	*	○	○	○
○	○	○	○	○	○	○
*	○	○	○	○	○	○

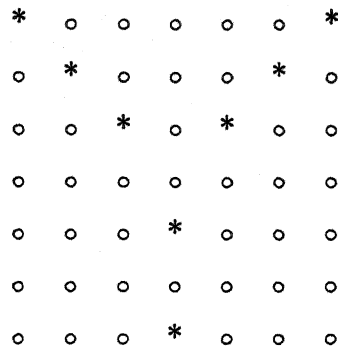
W

○	○	○	○	○	○	○
○	○	○	○	○	○	*
*	○	○	○	○	○	○
○	○	○	*	○	○	*
○	○	○	○	*	○	*
○	○	○	○	○	○	○
○	○	○	○	○	○	*

X

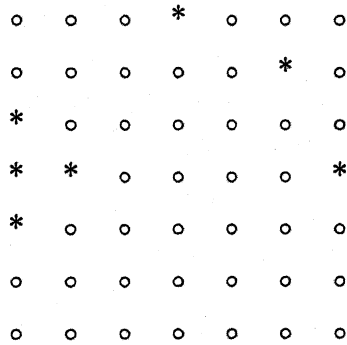
*	○	○	○	○	○	*
○	○	○	○	○	○	○
○	○	○	○	○	○	○
○	○	○	○	○	○	○
○	○	○	○	○	○	○
○	*	○	○	○	*	○
*	○	○	○	○	○	*

Y

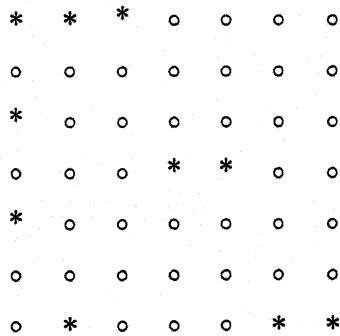


A.2.5 Adaptive Resonance Theory 1

A



E



I

○	*	○	○	○	*	○
○	○	○	○	○	○	○
○	○	○	*	○	○	○
○	○	○	○	○	○	○
○	○	○	*	○	○	○
○	○	○	○	○	○	○
○	*	○	○	○	*	○

L

○	○	○	○	○	○	○
*	○	○	○	○	○	○
○	○	○	○	○	○	○
*	○	○	○	○	○	○
○	○	○	○	○	○	○
○	○	○	○	○	○	○
○	○	○	○	○	○	○

M

○	○	○	○	○	○	○
*	○	○	○	○	○	*
*	○	*	○	○	○	*
*	○	○	○	○	○	○
○	○	○	○	○	○	○
○	○	○	○	○	○	○
○	○	○	○	○	○	○

X

*	○	○	○	○	○	*
○	○	○	○	○	○	○
○	○	○	○	○	○	○
○	○	○	○	○	○	○
○	○	*	○	*	○	○
○	○	○	○	○	○	○
○	○	○	○	○	○	○

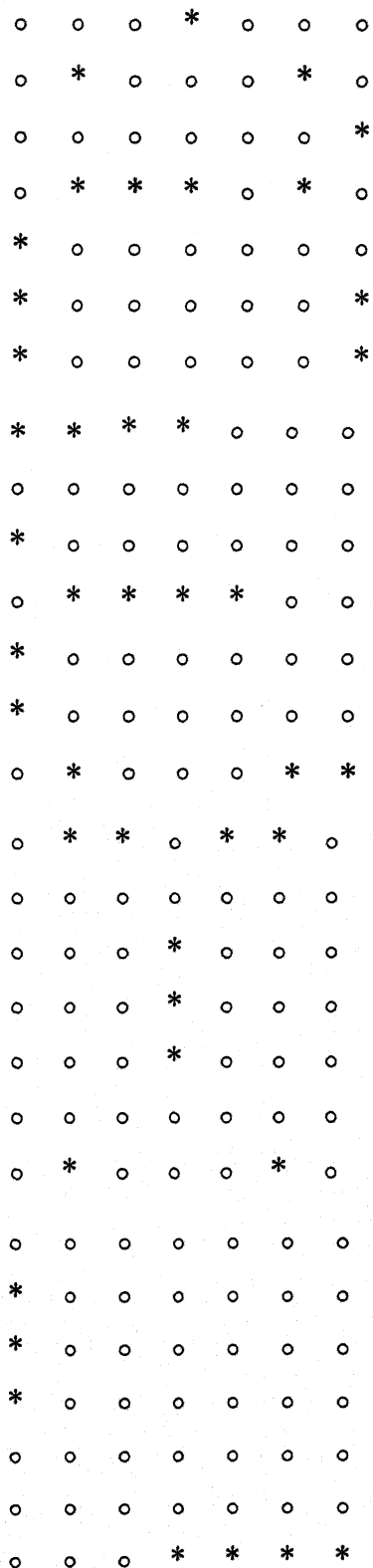
A.2.6 Kohonen SOM

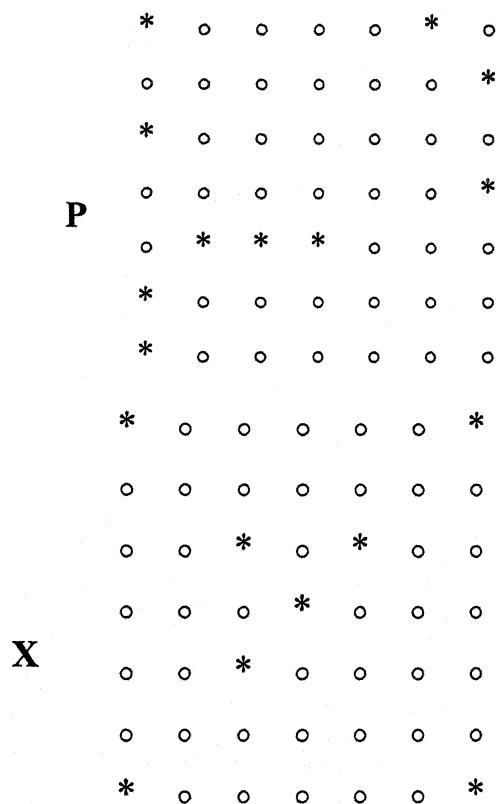
A

E

I

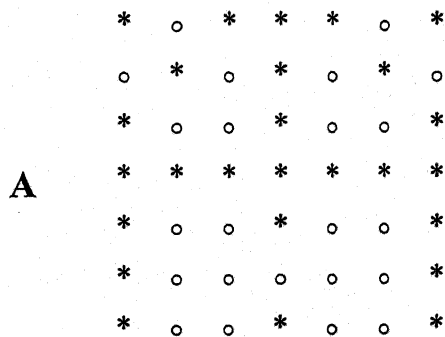
L





A:3 For Each Algorithm: Presentation of Characters After Adding Information

A.3.1 Bidirectional Associative Memory



I

○	*	*	*	*	*	○
*	○	○	*	○	○	○
*	*	○	*	*	*	○
○	*	*	*	○	*	*
*	○	○	*	○	○	○
○	○	○	*	○	○	○
○	*	*	*	*	*	○

L

*	○	○	○	○	○	*
*	○	○	○	○	*	○
*	○	○	○	*	○	○
*	○	○	○	○	○	○
*	○	○	○	○	*	*
*	○	○	○	○	○	○
*	*	*	*	*	*	*

M

*	○	*	*	*	○	*
*	*	○	○	○	*	*
*	○	*	*	*	○	*
*	○	○	*	○	○	*
*	○	○	*	○	○	*
*	○	*	*	*	○	*
*	○	*	*	*	○	*

P

*	*	*	*	*	*	○
*	○	○	*	*	○	*
*	*	○	○	○	○	*
*	*	○	○	*	○	*
*	*	*	*	*	*	○
*	*	○	○	○	*	○
*	○	○	*	*	○	*

X

*	o	*	*	*	o	*
o	*	o	*	o	*	o
o	o	*	*	*	o	o
o	o	o	*	o	o	o
o	o	*	*	*	o	o
o	*	o	o	o	*	o
*	o	*	*	*	o	*

A.3.2 Hopfield Auto Associative Memory

I

o	*	*	*	*	*	o
*	o	o	*	o	o	o
*	*	o	*	*	*	o
o	*	*	*	o	*	*
*	o	o	*	o	o	*
o	o	o	*	o	o	o
o	*	*	*	*	*	o
*	o	o	o	o	o	*
*	o	o	o	o	*	o
*	o	o	o	*	o	o
*	o	o	o	o	o	o
*	o	o	o	o	*	*
*	o	o	o	o	o	o
*	*	*	*	*	*	*

L

M

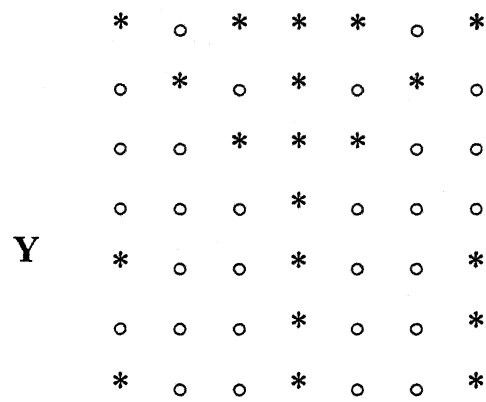
*	○	*	*	*	○	*
*	*	○	*	○	*	*
*	○	*	*	*	○	*
*	○	*	*	*	○	*
*	○	○	*	○	○	*
*	○	*	*	*	○	*
*	○	*	*	*	○	*

O

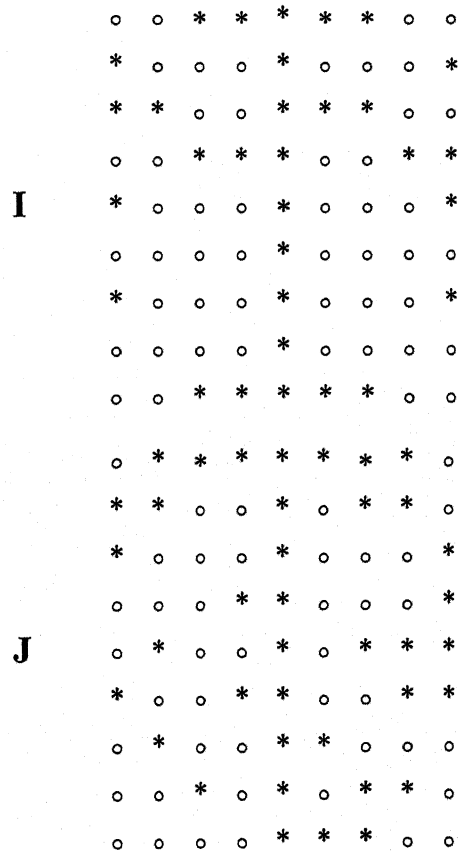
○	*	*	*	*	*	○
*	*	○	○	○	○	*
*	○	*	○	○	○	*
*	○	○	*	○	○	*
*	○	○	○	*	○	*
*	○	○	○	○	*	*
○	*	*	*	*	*	○

T

*	*	*	*	*	*	*
*	*	○	*	○	*	*
○	○	○	*	○	○	○
○	○	○	*	*	○	○
○	○	*	*	○	○	○
○	○	○	*	*	○	○
○	○	○	*	○	○	○



A.3.3 Feature Recognition Neural Network



L

```

*  o  o  o  o  o  o  o  *
*  o  o  o  o  o  o  *  o
*  o  o  o  o  o  *  o  o
*  o  o  o  o  o  o  o  o
*  o  o  o  o  o  o  o  *
*  o  o  o  o  o  o  o  o
*  o  o  o  o  o  o  o  *
*  o  o  o  o  o  o  o  o
*  *  *  *  *  *  *  *  *

```

M

```

*  o  *  *  *  *  *  o  *
*  *  o  *  *  *  o  *  *
*  o  *  *  *  *  *  o  *
*  *  o  *  o  *  o  o  *
*  o  o  o  *  *  o  o  *
*  o  o  o  o  o  o  o  *
*  o  o  o  *  o  o  o  *
*  o  o  *  *  *  o  o  *
*  o  o  *  *  *  o  o  *

```

P

```

*  *  *  *  *  *  *  o
*  o  o  *  *  *  *  o  *
*  o  *  *  *  o  o  *  *
*  *  o  *  *  o  *  o  *
*  *  *  *  *  *  *  o
*  *  o  *  *  o  *  *  o
*  o  o  *  *  *  o  o  *
*  o  o  *  *  *  *  o  o
*  o  o  *  *  *  o  *  o

```


Y

*	o	*	*	*	*	*	o	*
o	*	o	*	*	*	o	*	o
o	o	*	o	*	o	*	o	o
o	o	o	*	*	*	o	o	o
o	o	o	o	*	o	o	o	o
*	o	o	o	*	o	o	o	*
o	o	o	o	*	o	o	o	o
*	o	o	o	*	o	o	o	*
o	o	o	o	*	o	o	o	o

A.3.4 Hamming and MAXNET

I

o	*	*	*	*	*	o
o	o	o	*	o	o	*
*	*	o	*	*	*	o
o	*	*	*	o	*	*
*	o	o	*	o	o	*
o	o	o	*	o	o	*
o	*	*	*	*	*	o
o	*	*	*	*	*	o
*	o	*	*	o	*	*
*	*	o	*	*	o	*
*	o	*	*	o	*	*
o	*	o	*	o	*	*
o	o	*	*	o	*	*
*	*	o	*	*	o	*

J

M

*	○	*	*	*	○	*
*	*	○	*	○	*	*
*	○	*	*	*	○	*
*	○	*	*	*	○	*
*	○	○	*	○	○	*
*	○	*	*	*	○	*
*	○	*	*	*	○	*

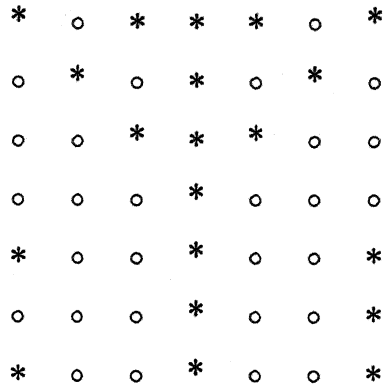
P

*	*	*	*	*	*	○
*	○	*	*	*	○	*
*	*	○	○	*	*	*
*	*	*	○	*	*	*
*	*	*	*	*	*	○
*	*	○	*	○	*	*
*	○	*	*	*	○	*

U

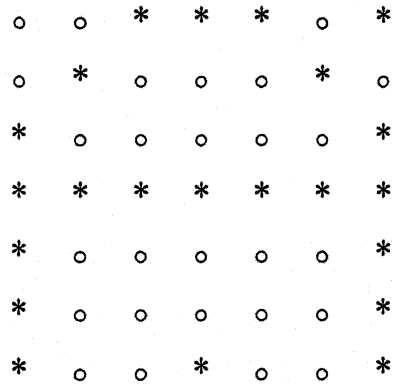
*	○	*	*	*	○	*
*	*	*	○	○	*	*
*	*	○	*	*	○	*
*	*	○	*	*	*	*
*	*	○	*	*	○	*
○	*	○	*	○	*	○
*	○	*	*	*	*	○

Y

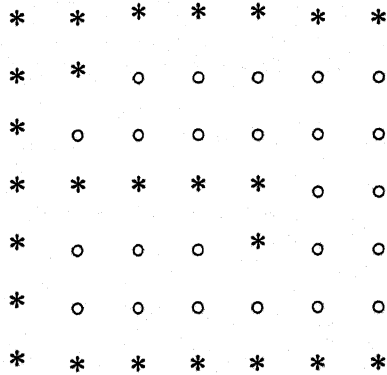


A.3.5 Adaptive Resonance Theory 1

A



E



I

*	*	*	*	*	*	*
*	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	o
o	o	o	*	o	o	*
*	*	*	*	*	*	*

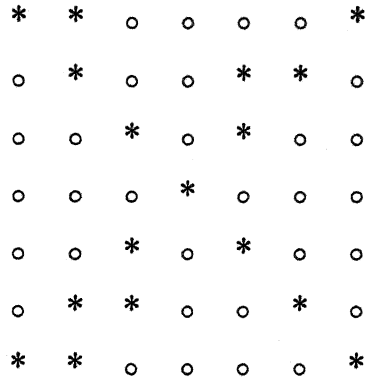
L

*	*	o	o	o	o	o
*	*	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	o	o	o	o	o	o
*	*	*	*	*	*	*

M

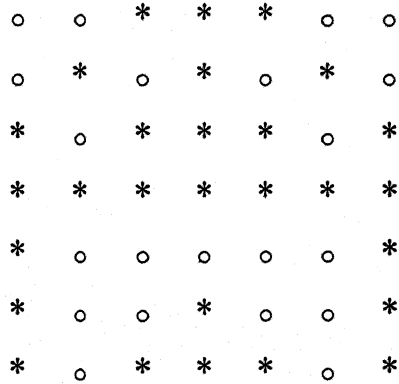
*	o	o	o	o	o	*
*	*	o	o	o	*	*
*	o	*	o	*	o	*
*	o	o	*	o	o	*
*	o	o	*	o	o	*
*	o	*	*	*	o	*
*	o	*	*	*	o	*

X

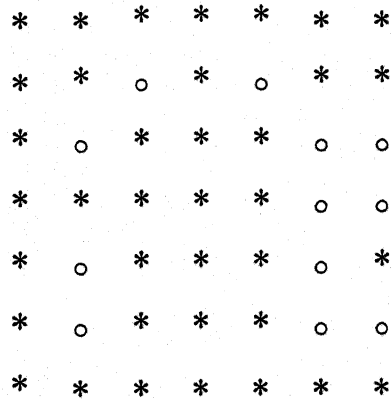


A.3.6 Kohonen SOM

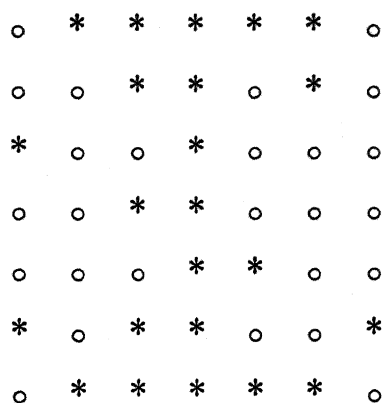
A



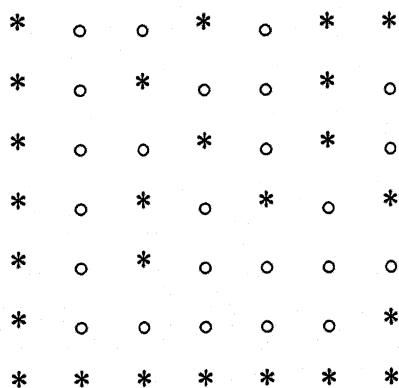
E



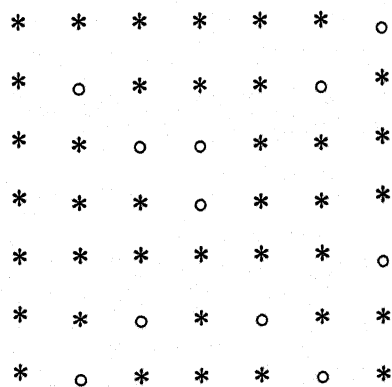
I



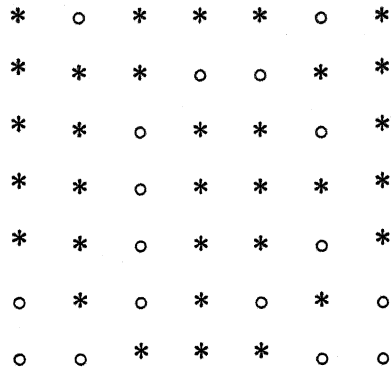
L



P



U



A.1: Representation of Input pattern in each algorithm

The size of pattern grid is taken as 7×7 in 6 algorithms. In FRNN, back propagation and Neocognitron it is taken as 9×9 , 7×5 and 19×19 respectively. Table A.1 shows a detail description about the size of pattern grids and the way in which on and off pixels are read in different algorithms.

Algorithms	Size of pattern Grid	Off pixels	On pixels
Bam	7×7	-1	1
Hopfield Auto associative Memory	7×7	-1	1
FRNN	9×9	-1	1
Hamming Network And MAXNET	7×7	-1	1
Backpropagation Algorithm	7×5	0	1
Quickprop	7×7	0	1
Art1	7×7	0	1

Kohonen SOM	7×7	0	1
Neocognitron	19×19	0	1

Table A.1: For each algorithm: Size of pattern grid and the way in which an on and off pixel read as

References

1. Magnetic ink character recognition system with angled read head IBM technical disclosure Bull. 28, 2555-2556 (1985).
2. J. Mantas, An overview of character recognition methodologies, pattern recognition **19**, 425-439 (1986).
3. Character Recognition. British Computer Society, London, England (1971).
4. K.S.Fu, Syntactic Pattern recognition and applications Prentice Hall, Eaglewood Cliffs, New Jersey.(1982).
5. G.Nagy, optical character recognition: theory and practice, handbooks of statistics, P.R. Krishnaiah and L.N.Kanal, Eds, Vol.2,pp.621-649 (1982).
6. S.N. Srihari, computer text recognition and error correction. IEEE computer society press, silver spring, MD (1984)
7. J.R.Ullmann, advances in character recognition, application of pattern recognition, K.S.Fu, Ed., pp.197-236. CRC Press, Boca Raton, FL, 1982.
8. V.A. Kovalevisky, Ed., Character readers and pattern recognition, Spartan books, New York, 1968.
9. Y.A. Kovalevisky, Image pattern recognition, Springer, Berlin (1977).
10. Optical character recognition and years ahead. The Business press, Illinois, USA, 1969.
11. C.Y. Suen and R.D.Mori, Eds, Computer analysis and perception, Vol. I: Visual signals. CRC press, Boca Raton, FL, 1982.
12. G.C Smitch, The stereo toner reading aid for the blind, a progress report, 1973 Carnahan Conf. on electronic prosthetics, Lexington, USA, pp. 74-76 (19-21 Sept. 1973).

- 13.R.D. Badoux, DELTA [Text reader for the blind], computerized Braille production, Proc. 5th int. workshop, Winterthur, Switzerland, pp.21-25 (30 October-1 November 1985).
- 14.G.V.Kondraske and A. Shennib, character pattern recognition for a telecommunication aid for the deaf, IEEE T. Biomed.Engng.**33**,366-370 (1986).
- 15.C.W.Swonger, An evaluation of character normalization, feature extraction and classification techniques for postal mail reading, proc. Automatic Pattern Recognition, Washington, D.C., USA.,pp.67-87, 6th may 1969.
- 16.H. Genchi, S.Watanabe, S.Matsunaga and M.Tamada, automatic reader-shorter for mail with hand written or printed postal code numbers, Toshiba Rev. (int. edn.) Japan, **49**, 7-11 (1970).
- 17.K.Notbohm and W.Hanisch, Automatic Digit recognition in a mail sorting machine, Nachrichtentech, electron, (Germany)**36**, 472-476 (1986) (In German).
- 18.G.L. Skalski, OCR in the publishing industry, Data processing XII, Proc., 1967, Int.Data process. Conf. And business exposition, Boston, MA, USA.,pp.255-260 (20-23, June 1967).
- 19.H.Genchi, Data communication terminal apparatus, optical character and mark readers, Denshi Tsushin Gakkai Zasshi, **52**,418-428 (1969) (In Japanese).
- 20.J.Ufer, Direct data processing with the IBM 1287 multipurpose document reader for standard article-fresh service to Joh. Jacob and Co. Breman, IBM Nachr. (Germany) **20**, 35-40 (February 1970). (In German).
- 21.M.Vossen, Electronic page reader in use, office management (Germany) **34**, 1148 (1986). (In German).

- 22.K. Yoshida, optical character reader for telephone exchange charge billing system, Japan telecom. rev. (Japan) **16**, 105-110 (1974).
- 23.G.Hilgert, Method of dealing with orders on the IBM 1287 multifunction document reader at the decentralised sales organisation of the continental Gummi-Werke Aktiengesellschaft, IBM Nachr. **20**, 122-125 (1970). (In German).
24. Automatic Identification of latent finger-prints, report (unnumbered), (PB-192976), Metropolitan Atlanta council of local government, GA, USA, P.31 (April 1970).
- 25.W.Bojman, Detection and/or measurement on complex patterns, IBM technical disclosure Bull. (USA) **13**, 1429-1430 (1970).
26. Field oriented Scanning system, IBM Technical disclosure Bull. (USA) **29**, 2130-2133 (Oct. 1986).
- 27.F.H. Murphy and E.A. Stohr, Optimal Check sorting strategies, Bull. Oper. Res. Am. **23** (supplement 1), B/145 (Spring 1975).
- 28.S.Kupriyanov, electronic handwriting analyzer, Tekh. Misul (Bulgaria) **9**, 7-13 (1972) (In Bulgarian).
- 29.J. Sternberg, automated signature verification using handwriting pressure, 1975 WOSCON Technical papers-western electronic Show and Convention, **19**, San Francisco, California, USA, 31-34 (16-19 sept, 1975).
30. Schalkoff R.J., "Pattern Recognition: Statistical Structured and Neural; Approach", John Wiley and Sons, 1992
31. Fukuanga K., "Statistical Pattern recognition-2nd ed.", Academic Press, 1990.
32. Fu K.S., "Syntactic Pattern Recognition," Prentice Hall, 1982.

33. Gonzalez R.C. and Thomason M.G., "syntactic Pattern Recognition", Addison-wesley, 1978.
34. Chen.C.H.Ed., " Proceeding of the IEEE workshop on expert system and pattern analysis" world scientific,"1987.
35. Hush D.R. and Harn B.G., "Progress in supervised Neural Networks: What's new Since Lippmann", IEEE signal processing magazine, January 1993, pp.8-39.
36. V.K.Govindan and A.P. Sivaprasad, "Character Recognition- A review", Pattern recognition, Vol.23, No. 7, PP. 671-683, 1990.
37. Arbib, M.A.1987. Brains, Machines and Mathematics, 2nd Ed. New York: Springer Verlag.
38. Carpenter, G.A. 1989. "Neural Network Models for pattern recognition and associative memory," Neural Network 2: 243-257.
39. Feldman, J. A., M.A. Fanty, and N. Goddard. 1988. "Computing with structured Neural Networks," IEEE Computer (March):91-103.
40. T. Kohonen, *Self Organization & Associative Memory*. Berlin: Springer Verlag, 1984.
41. Hebb D.O., 1961. Organization of Behavior, NewYork: Science Edition.
42. Hongchi Shi, Yunxin zhao and Xinhua Zhuang," A general Model for Bidirectional Associative memory", IEEE Transactions on systems, Man and cybernetics Part B: Cybernetics, Vol. 28, No. 4, PP. 511-518, August 1998.
43. Kosoko, B. 1987."Addaptive Bidirection Associative Memories," Appl. Opt. 26 (23): 4947-4959.
44. Kosoko, B. 1988."Addaptive Bidirection Associative Memories," IEEE Trans. Systems, Man, and Cybernetics 18(1): 49-60.

45. Z.B. Xu, Y. Leung and X.W. He, "Asymmetrical Bidirectional Associative Memories", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24, PP. 1558-1564, Oct. 1994.
46. J. J. Hopfield and D. W. Tank, "Computing with neural circuits: A model," *Science*, vol. 233, pp. 625-633, Aug. 1986.
47. J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons." *Proc. Nat. Acad. Sci.*, vol. 81, no. 10, pp. 3088-3092, May 1984.
48. Symon Haikin, "Neural Networks: A comprehensive Foundation", Macmillan College Publishing Company, New York, 1994.
49. Basit Hussain and M.R. Kabuka, "A Novel Feature Recognition Neural Network and its Application to character Recognition", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 16, No. 1, PP. 98-106, Jan. 1994.
50. Jacek M. Zurada, "Introduction to artificial Neural Systems", Jaico Publication House.
51. Carpenter, G.A. and S. Grossberg. 1987. "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics and Image Proc.* 37: 54-115.
52. Carpenter, G.A. and S. Grossberg. 1988. "Neural Dynamics of Category Learning and Recognition: Attention, Manual Consolidation and Amnesia" In *Brain Structure, Learning and Memory* Ed. J. Davis, R. Nubergh, I. Wegman. AAAS symp. Series. Westview press, Boulder, Colo.
53. P. Werbos, "Backpropagation: Past and future," in *Proc. IEEE Inf. Conf. Neural Networks*, San Diego, CA, July 1988.

- 54.M. L. Brady, R. Raghavan, and I. Slawny, "Back propagation fails to separate where perceptions succeed," *IEEE Trans. Circuits Syst.*, vol. 36, no. 5, pp. 665-674, May 1989.
- 55.Kohonen T., 1988. "The 'Neural' Phonetic Typewriter," *IEEE Computer* 27(3): 11-22.
- 56.K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 82-34, Sept. 1983.
- 57.K. Fukushima and N. Wake, "Handwritten alphanumeric character recognition by the Neocognitron," *IEEE Trans. Neural Networks*, vol.2, no. 3, May 1991.
- 58.K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in positions," *Pattern Recognition*, vol. 15, no. 6, pp. 455-469, 1982.
- 59.J. I. Minnix, E. S. McVey, and R. M. Ifiigo, "Modified Neocognitron with position normalizing preprocessor for translation invariant shape recognition," *Int. Joint Conf. Neural Nets*, vol. 1, June 1990, pp. 395-399.
- 60.C. Sung and D. Wilson, "Precognition: Neocognitron coupled with perceptron," *Int. Joint Conf Neural Nets*, vol. 3, June 1990, pp. 753-758.